UNIVERSIDAD NACIONAL DE CHIMBORAZO

FACULTAD DE INGENIERÍA
CARRERA INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN

Sistema para la gestión de censo de salud en el centro de salud Chambo utilizando el framework Flutter

Trabajo de Titulación para optar al título de Ingeniero en Tecnologías de la Información

Autor:
Logroño Casco Anthony Alexander
Silva Silva Michelle Carolina

Tutor:
Ing. Danny Patricio Velasco Silva

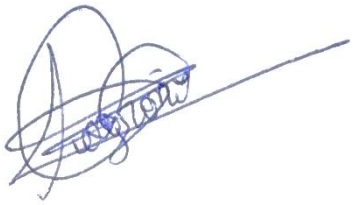
Riobamba, Ecuador. 2026

DECLARATORIA DE AUTORÍA

Nosotros, Anthony Alexander Logroño Casco, con cédula de ciudadanía 0605936624, y Michelle Carolina Silva Silva con cédula de ciudadanía 0650302334, autores del trabajo de investigación titulado: Sistema para la gestión de censo de salud en el centro de salud Chambo utilizando el framework Flutter, certifico que la producción, ideas, opiniones, criterios, contenidos y conclusiones expuestas son de mí exclusiva responsabilidad.

Asimismo, cedo a la Universidad Nacional de Chimborazo, en forma no exclusiva, los derechos para su uso, comunicación pública, distribución, divulgación y/o reproducción total o parcial, por medio físico o digital; en esta cesión se entiende que el cesionario no podrá obtener beneficios económicos. La posible reclamación de terceros respecto de los derechos de autor (a) de la obra referida, será de mi entera responsabilidad; librando a la Universidad Nacional de Chimborazo de posibles obligaciones.

En Riobamba, 08 de octubre del 2025.



Anthony Alexander Logroño Casco

C.I: 0605936624

Michelle Carolina Silva Silva

C.I: 0650302334

DICTAMEN FAVORABLE DEL PROFESOR TUTOR

Quien suscribe, Danny Patricio Velasco Silva catedrático adscrito a la Facultad de Ingeniería, por medio del presente documento certifico haber asesorado y revisado el desarrollo del trabajo de investigación titulado: SISTEMA PARA LA GESTIÓN DE CENSO DE SALUD EN EL CENTRO DE SALUD CHAMBO UTILIZANDO EL FRAMEWORK FLUTTER, bajo la autoría de Anthony Alexander Logroño Casco y Michelle Carolina Silva Silva; por lo que se autoriza ejecutar los trámites legales para su sustentación.

Es todo cuanto informar en honor a la verdad; en Riobamba, a los 08 días del mes de octubre de 2025

Firmado electrónicamente por:
DANNY PATRICIO
VELASCO SILVA
Validar únicamente con FirmaEC

Danny Patricio Velasco Silva

C.I: 0602768640

CERTIFICADO DE LOS MIEMBROS DEL TRIBUNAL

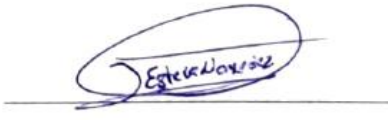
Quienes suscribimos, catedráticos designados Miembros del Tribunal de Grado para la evaluación del trabajo de investigación "SISTEMA PARA LA GESTIÓN DE CENSO DE SALUD EN EL CENTRO DE SALUD CHAMBO UTILIZANDO EL FRAMEWORK FLUTTER" por Anthony Alexander Logroño Casco con cédula de identidad número 0605936624 y Michelle Carolina Silva Silva con cédula de identidad número 0650302334, bajo la tutoría de Ing. Danny Velasco; certificamos que recomendamos la APROBACIÓN de este con fines de titulación. Previamente se ha evaluado el trabajo de investigación y escuchada la sustentación por parte de su autor; no teniendo más nada que observar.

De conformidad a la normativa aplicable firmamos, en Riobamba 16 de diciembre del 2025.

Pamela Buñay, Mg.
PRESIDENTE DEL TRIBUNAL DE GRADO

Ana Congacha, Mg.
MIEMBRO DEL TRIBUNAL DE GRADO

Miryan Narváez, PhD.
MIEMBRO DEL TRIBUNAL DE GRADO



C E R T I F I C A C I Ó N

Que, **LOGROÑO CASCO ANTHONY ALEXANDER** con CC: **0605936624 y SILVA SILVA MICHELLE CAROLINA** con CC: **0650302334**, estudiantes de la Carrera **INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN**, Facultad de **INGENIERÍA**; han trabajado bajo mi tutoría el trabajo de investigación titulado **SISTEMA PARA LA GESTIÓN DE CENSO DE SALUD EN EL CENTRO DE SALUD CHAMBO UTILIZANDO EL FRAMEWORK FLUTTER**, cumple con el 2%, de acuerdo al reporte del sistema Anti plagio **COMPILATIO**, porcentaje aceptado de acuerdo a la reglamentación institucional, por consiguiente autorizo continuar con el proceso.

Riobamba, 22 de noviembre de 2025

Firmado electrónicamente por:
DANNY PATRICIO
VELASCO SILVA
Validar únicamente con FirmaEC

Mgs. Danny Velasco Silva
TUTOR

DEDICATORIA

Dedico este trabajo de investigación, en primer lugar, a mis padres, quienes han sido mi mayor fuente de inspiración y apoyo incondicional. A mi madre, por su amor infinito, su fortaleza y por estar siempre presente, brindándome su apoyo tanto emocional como económico. A mi padre, por ser ese pilar firme que nunca me dejó solo, acompañándome en cada paso, en cada desafío, y por motivarme siempre a dar lo mejor de mí en cada reto académico.

Quiero también dedicar esta tesis a mis abuelitos, aquellos que ahora me cuidan desde el cielo, Papá Lucho y Mamá Tina, que siempre quisieron verme convertido en un profesional, y a mis abuelos que siguen conmigo, Mamá Trini y Papá Pablo, cuyo amor y apoyo incondicional han sido fundamentales en este camino. A Mamá Trini, en especial, por haberme criado desde pequeño y por enseñarme con su ejemplo el valor de la dedicación y el amor.

No puedo dejar de mencionar a mis queridas mascotas, que, aunque no comprendan el significado de este logro, fueron un sostén emocional invaluable en los momentos difíciles.

A toda mi familia, gracias por ser parte de este sueño hecho realidad. Esta tesis lleva impreso el esfuerzo, el amor y el apoyo que he recibido de cada uno de ustedes.

Anthony Logroño

Dedico este trabajo de investigación a mi madre, por ser esa mujer increíble que me enseñó con su ejemplo lo que significa la fortaleza, el amor incondicional y la entrega total. Eres y siempre serás mi mayor inspiración, gracias por cada palabra de aliento, por cada sacrificio y por nunca soltar mi mano, incluso cuando yo misma dudaba. A quien me ha acompañado con amor genuino y constante, mi amado novio, gracias por estar cuando más te necesitaba, por impulsarme a seguir adelante cuando sentía que no podía más y por abrazar cada una de mis versiones, incluso las más cansadas. Tu paciencia, tu fe en mí y tu amor constante han sido luz en los días oscuros y fuerza en los días débiles. No hubiera llegado tan lejos sin ti.

A mi querido amigo Francisco Navarrete, gracias por ser esa amistad firme y verdadera, por las largas noches de estudio, por compartir el cansancio sin quejarte, por acompañarme en silencio y también en risa. Tu apoyo ha sido compañía real y profunda, de esas que se sienten incluso cuando no se dicen. A mis docentes, a quienes respeto profundamente, gracias por ser guías pacientes y por sembrar no solo conocimiento, sino pasión, disciplina y confianza. Cada palabra y enseñanza suya ha dejado huella en mi camino. A todos ustedes, por sostenerme, por creer en mí y por caminar a mi lado. Esta tesis es el fruto de un esfuerzo compartido, y en cada página vive un pedacito de su amor, su entrega y su apoyo.

Michelle Silva

AGRADECIMIENTO

Agradezco, ante todo, a mis padres, Diego Logroño y Maribel Casco, por haberme dado siempre su apoyo incondicional y enseñarme, con su ejemplo, que los sueños se alcanzan con esfuerzo, sacrificio y amor. Gracias por ser la base sobre la cual he podido construir cada meta.
A mis abuelos, Papá Lucho y Mamá Tina, que, aunque ya no están, siempre permanecen en mi memoria y en mis logros, y a Mamá Trini y Papá Pablo, por su cariño y respaldo incondicional.
A mis tíos, Luis Logroño y Silvia Casco, por haberme acogido en una etapa importante de mi vida y brindarme su apoyo como si fueran unos segundos padres. Y a mi primo Josué, por ser más que un primo, un verdadero hermano y compañero en los momentos difíciles.
A Michelle Silva, por haber sido mi compañera en este camino, por su disposición para ayudarme siempre que lo necesité y por motivarme a seguir adelante, incluso en los momentos más difíciles.
También agradezco a mis docentes y a mi tutor, por sus enseñanzas y por transmitirme el compromiso y la pasión por esta profesión.
A todas las personas que, de distintas maneras, han sido parte de este camino, gracias por su apoyo y por contribuir a la realización de este logro.

Anthony Logroño

Agradezco a Dios por llenarme de sabiduría cuando más la necesité, por darme serenidad en medio del cansancio y por fortalecer mi espíritu en cada momento de duda. Su presencia ha sido mi guía silenciosa y constante, dándome fuerzas cuando sentía que ya no podía más.
A mi madre, gracias por ser mi apoyo más incondicional, por tus palabras firmes cuando necesitaba ánimo y por estar presente en cada paso, aún en los más difíciles. Eres el corazón que me sostuvo y la fuerza que me impulsó a no rendirme.
A mis amigos Dennis, Felipe y Anthony, que se convirtieron en mi segunda familia, gracias por cada gesto, cada palabra, cada desvelo compartido. Su compañía sincera, su paciencia y su lealtad hicieron que este camino fuera más llevadero y lleno de aprendizajes. Gracias por estar, por quedarse y por creer en mí.
A mi novio, por estar a mi lado durante todo el semestre, por ayudarme en cada tarea, por acompañarme en los momentos de estrés y por no soltarme nunca. Tu apoyo, tu paciencia y tu amor hicieron toda la diferencia en este proceso.

Michelle Silva

ÍNDICE GENERAL

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

RESUMEN

La presente investigación abordó el desarrollo de un sistema multiplataforma para la gestión del censo de salud en el Centro de Salud Chambo, empleando el framework Flutter para la aplicación móvil, mientras que para el apartado web se utilizó Node.js para el backend y React para el frontend. El desarrollo del sistema se estructuró en fases bajo la metodología ágil Mobile-D, permitiendo implementar un proceso iterativo e incremental, garantizando la integración de funcionalidades tanto en la aplicación móvil como en la plataforma web.

El proceso de desarrollo comenzó con la fase de exploración, en la cual se identificaron los requisitos funcionales y no funcionales, los grupos de interés y el alcance del sistema; en la fase de inicialización se configuró el entorno de desarrollo, se diseñaron los diagramas de base de datos, casos de uso y esquemas de navegación; mientras que en la fase de producción y estabilización se llevó a cabo el desarrollo integral de las plataformas móvil y web, integrando funcionalidades como autenticación y validación de usuarios por roles, apartado offline, carga masiva de datos en formato .xlxs, gestión de usuarios, registro de encuestas de salud, campañas de vacunación y más características que se adaptaron a las necesidades del Centro de Salud Chambo.

Finalmente, en la fase de pruebas del sistema se planificaron y ejecutaron simulaciones basadas en la norma ISO/IEC 25012:2008 para medir la eficiencia, precisión y disponibilidad de los datos. Los resultados obtenidos gracias a estas pruebas reflejaron una elevada precisión en la captura y almacenamiento de los registros de los usuarios, evitando perdidas o inconsistencias, además, confirmaron una buena disponibilidad, asegurando que la información esté accesible de manera consiste y sin retrasos.

Palabras claves: analítica de datos, aplicación híbrida, aplicación offline, censo de salud, fiabilidad de los datos, Flutter, sistema de gestión, sincronización de datos.

ABSTRACT

This research addressed the development of a multiplatform system for managing the health census at the Chambo Health Center, using the Flutter framework for the mobile application, while Node.js was used for the backend and React for the frontend of the web platform. The system development was organized into phases following the Mobile-D agile methodology, which enabled an iterative and incremental process and ensured the integration of functionalities in both the mobile application and the web platform.

The development process began with the exploration phase, in which the functional and non-functional requirements, stakeholders, and system scope were identified. In the initialization phase, the development environment was configured, and the database diagrams, use cases, and navigation schemas were designed. Subsequently, during the production and stabilization phases, the comprehensive development of both platforms was carried out, incorporating functionalities such as user authentication and role-based validation, offline mode, bulk data upload in .xlsx format, user management, health survey registration, vaccination campaign management, and other features tailored to the needs of the Chambo Health Center.

Finally, during the testing phase, evaluations were planned and executed based on the ISO/IEC 25012:2008 standard to assess data efficiency, accuracy, and availability. The results demonstrated high accuracy in capturing and storing records, preventing data loss or inconsistencies; additionally, they confirmed adequate availability, ensuring that information remains consistently accessible without delays.

Keywords: data analytics, data reliability, data synchronization, Flutter, health census, hybrid application, management system, offline application.

Firmado electrónicamente por:
LOURDES DEL ROCIO
QUINATA ENCARNACION
Validar únicamente con FirmaEC

Reviewed by:

Mg. Lourdes del Rocío Quinata Encarnación

ENGLISH PROFESSOR

C.C 1803476215

CAPÍTULO I. INTRODUCCIÓN

En la actualidad, la gestión eficiente de la información sanitaria es un desafío crítico para los sistemas de salud pública a nivel mundial. La capacidad de recopilar, organizar y analizar datos de manera efectiva se ha convertido en un elemento esencial para mejorar la calidad de los servicios y garantizar una planificación estratégica basada en evidencia. Según la Organización Mundial de la Salud [1], la implementación de sistemas tecnológicos en el ámbito sanitario permite una mejora significativa en la accesibilidad, precisión y utilización de los datos, aspecto que resulta clave para responder a las crecientes demandas de los servicios de salud.

El Centro de Salud de Chambo, ubicado en una región rural del Ecuador, no es ajeno a estos retos. Su dependencia de métodos manuales para la gestión de censos de salud, como fichas en papel y formularios impresos, ha limitado severamente su capacidad operativa. Este enfoque tradicional no solo incrementa la probabilidad de errores humanos, sino que también dificulta el acceso oportuno a información crítica. De acuerdo con el Ministerio de Salud Pública del Ecuador [2], ciertos centros de salud rurales enfrentaron dificultades similares, donde se señaló la urgente necesidad de modernizar sus procesos de gestión.

Por otra parte, el presente proyecto propone como objetivo diseñar e implementar un sistema de gestión de censos de salud para el Centro de Salud de Chambo, empleando tecnologías modernas que permitirán desarrollar un sistema multiplataforma destinado a optimizar la recolección y almacenamiento de información. El aplicativo móvil contará con módulos enfocados en distintas áreas de la salud como alimentación y nutrición, vacunación, tamizaje visual, salud oral, higiene y saneamiento, y salud mental. Además, el desarrollo de un sitio web complementario permitirá visualizar la información recolectada y ayudará en el análisis de los datos para una posterior toma de decisiones.

1.1 Planteamiento del problema

El Centro de Salud Chambo enfrenta dificultades importantes en la gestión del censo de salud debido a la dependencia de procesos manuales para la recopilación, procesamiento y análisis de datos. Este enfoque tradicional incrementa la probabilidad de errores humanos, duplicidad de registros, pérdida de información y falta de consistencia en los datos, afectando la fiabilidad de la información necesaria para la toma de decisiones oportunas y precisas.

Además, la ausencia de herramientas tecnológicas que optimicen la recopilación y validación de los datos limita la eficiencia y la capacidad de analizar grandes volúmenes de información de manera oportuna. Según el Ministerio de salud pública [3], el 76% de los sistemas informáticos del MSP son obsoletos y carecen de interoperabilidad. Esta situación repercute negativamente en la capacidad del personal médico para planificar y ejecutar estrategias de salud basadas en datos confiables, impactando directamente en la calidad del servicio ofrecido a la población.

Por otra parte, pero no menos importante, uno de los principales obstáculos en materia de conectividad es el acceso a internet en las zonas rurales. Según el Ministerio de salud pública [3], en el año 2023 solo el 44,4% de los hogares rurales contaban con conexión a internet, ya sea fija o móvil. Esta limitación puede influir negativamente el funcionamiento de un sistema digital, considerando que, en la mayoría de las ocasiones el almacenamiento de la información requiere de conexión a internet.

En este contexto, se plantea la necesidad de implementar un sistema para la gestión del censo de salud que permita optimizar la recopilación de información de manera offline, organización y análisis de los datos, evaluando su fiabilidad conforme a los criterios establecidos en la norma ISO/IEC 25012:2008, con un enfoque en la eficiencia, precisión y disponibilidad de los datos.

1.2 Justificación

El Ministerio de salud pública [3], en su plan de actualización de infraestructura tecnológica y fortalecimiento de conectividad en establecimientos de salud, busca reducir la brecha tecnológica en instituciones públicas y privadas del 60% en 2024 al 25% en 2034. La implementación de un sistema digital permitirá no solo facilitar el análisis de los datos para identificar patrones y tendencias que puedan guiar políticas de salud más efectivas, sino también dotar al personal de herramientas modernas y ágiles, mejorando así la atención a la población, que podrá acceder a una atención más rápida y basada en información confiable.

Por otro lado, este proyecto se enfoca en el desarrollo de un sistema multiplataforma para optimizar la recolección y gestión de la información del censo en instituciones educativas, incluyendo aquellas ubicadas en zonas rurales. A través de herramientas de desarrollo de aplicativos, bases de datos y gestión de procesos, se busca mejorar la cobertura y precisión de la información recolectada, permitiendo una mejor toma de decisiones en al ámbito de la salud.

1.3 Formulación del problema

¿Cómo impactará la implementación de un sistema WEB y MÓVIL para la gestión del censo de salud en la fiabilidad de los datos del Centro de Salud Chambo, considerando los criterios de eficiencia, precisión y disponibilidad establecidos por la norma ISO/IEC 25012:2008?

1.4 Objetivos

Objetivo General

Implementar un sistema para la gestión de censo de salud en el Centro de Salud Chambo utilizando el framework Flutter.

Objetivos Específicos

- Investigar la metodología de desarrollo ágil Mobile-D, para la implementación de un sistema que integre componentes de datos enfocados en el Centro de Salud Chambo.
- Desarrollar una aplicación web y móvil para la recopilación y análisis de información de censo en el centro de salud Chambo.
- Evaluar la fiabilidad de los datos en el sistema de gestión de censo de salud conforme con los criterios establecidos por la norma ISO/IEC 25012:2008.

CAPÍTULO II. MARCO TEÓRICO

2.1 Sistemas de censo de salud

Un censo se basa en la recolección sistemática de datos, además de organizarlos y analizarlos con la finalidad de obtener información precisa y actualizada de una población [4]. Según Colwill y Poullis [5], el uso de sistemas censales en aplicaciones móviles para el área de salud ayuda a que la recolección de datos sea más rápida y precisa, además, facilita el acceso a zonas rurales de países en desarrollo, constituyendo un recurso sumamente valioso y útil para la investigación de la planificación sanitaria.

2.2 Desarrollo de aplicaciones móviles

Una aplicación móvil es un software hecho para ejecutarse en teléfonos inteligentes y tabletas, con el propósito de ofrecer funcionalidades específicas al usuario. Estas aplicaciones pueden desarrollarse para distintas plataformas, como Android o iOS, o también pueden ser multiplataforma, utilizando tecnologías como Flutter o React Native. [6]

2.2.1 Aplicaciones móviles nativas

Son aplicaciones específicas de una plataforma aprovechan al máximo las capacidades del dispositivo, dando como resultado una mejor experiencia de usuario. Algunos ejemplos de estas aplicaciones pueden ser: mensajería, redes sociales, geolocalización, reproductor de archivos multimedia y otras apps que facilitan el acceso a varias funcionalidades. Huaraca [7] menciona que las aplicaciones nativas de la plataforma no están estandarizadas, dando como resultado mayor tiempo de desarrollo.

2.3 Aplicaciones web

Es un sistema informático accesible a través de un navegador, permitiendo a los usuarios interactuar con diversas funcionalidades y servicios sin la necesidad de instalar softwares de terceros en su dispositivo. Su funcionalidad se basa en la combinación de tecnologías como HTML, CSS y JavaScript, además de permitir la programación con lenguajes como Python, PHP o Node.js. De acuerdo con Guaman y Yambay [8], las aplicaciones web pueden diseñarse para una gran variedad de usos, de las que más se usan comúnmente pueden ser correos electrónicos, tiendas en línea, redes sociales, entre otras muchas aplicaciones que se pueden acceder únicamente mediante un navegador.

2.4 Ecosistema de desarrollo de aplicaciones móviles

2.4.1 Android Studio

Es un entorno de desarrollo para crear aplicaciones Android, desarrollado por Google y basado en IntelliJ IDEA. Ofrece desde herramientas para interfaces, escribir código en lenguajes como Kotlin, Java, Flutter o C++, probar la app con emuladores de distintas versiones de Android y generar los instaladores tanto en formato APK como AAB. Vallejo [9], menciona que su sistema de emulación integrado permite a los desarrolladores visualizar los cambios que se van realizando en tiempo real, además de permitir comprobar la

aplicación en diferentes dispositivos móviles que tienen distintas configuraciones y resoluciones simultáneamente.

2.4.2 Sistema operativo Android

Es un sistema operativo para dispositivos móviles desarrollado por Google. El propósito de Android es promover el uso de un S.O de tipo abierto, gratuito y seguro, además de adaptarse a teléfonos inteligentes, tablets y televisores. Este sistema basado en Linux, incluye una versión de Java llamada Dalvik, facilitando el desarrollo de aplicaciones que aprovechan las características de los dispositivos de manera sencilla. Según Valdivieso [10], Android es un sistema operativo de código abierto a cargo de Handset Alliance liderado por Google, basado en el núcleo o kernel de Linux lo que permite a los desarrolladores aprovechar al máximo el uso de este S.O.

2.4.3 Arquitectura de Android

La arquitectura del sistema operativo Android consta de cuatro capas principales que hacen posible un buen funcionamiento en dispositivos móviles. En la base se ubica el núcleo de Linux, seguido por la capa de bibliotecas y el entorno de ejecución. Sobre esta se ubica el framework de aplicaciones, que proporciona las interfaces y servicios necesarios para el desarrollo. Finalmente, en la capa superior se encuentran las aplicaciones, que interactúan directamente con los usuarios. Esta estructura modular y jerárquica además de dividirse en cuatro capas, cuenta con una gran variedad de características como se muestra en la Figura 1.

Figura 1: Arquitectura de android.
Fuente: [11]

2.4.4 Flutter

Framework de desarrollo de aplicaciones móviles multiplataforma creado por Google, cuyo principal objetivo es la construcción de aplicaciones nativas para sistemas operativos iOS y Android utilizando una única base de código. Esta característica optimiza significativamente los recursos y el tiempo en el proceso de desarrollo, garantizando la creación de aplicaciones eficientes y consistentes [12]. De acuerdo con Baldrés [13], una de las principales ventajas de Flutter es su arquitectura basada en widgets, que permite a los desarrolladores diseñar

interfaces de usuario altamente personalizables y de alto rendimiento. Asimismo, Flutter emplea el lenguaje de programación Dart, también desarrollado por Google, especialmente optimizado para facilitar la creación de interfaces de usuario reactivas y eficientes.

En cuanto a su arquitectura, Flutter adopta un modelo basado en capas, donde el framework proporciona herramientas para la gestión de widgets, gestos y animaciones, mientras que la capa del motor se encarga del renderizado gráfico y la comunicación con la plataforma del dispositivo, como se detalla en la Figura 2. Esta estructura modular mejora la escalabilidad de las aplicaciones y optimiza la integración con servicios externos, como bases de datos y autenticación.

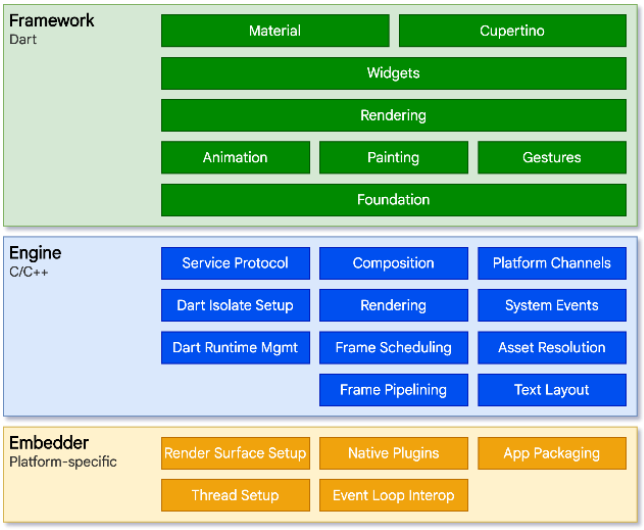


Figura 2: Capas arquitectónicas de Flutter.
Fuente: [14]

2.4.5 Dart

Es un nuevo lenguaje de programación orientado a objetos y desarrollado por la empresa Google, donde su uso se enfoca principalmente en crear aplicaciones móviles y web, además de ser el lenguaje base del framework Flutter desarrollado por la misma empresa. Según Escobar [15], Dart es conocido por su gran flexibilidad y por ser un lenguaje de código abierto. Su diseño optimizado hace posible que Dart se ejecute fluidamente gracias a su compilación Just-In-Time y Ahead-Of-Time, permitiendo a los desarrolladores crear aplicaciones agiles y responsivas.

2.5 Ecosistema de desarrollo de aplicaciones web

2.5.1 React.js

Biblioteca de JavaScript desarrollada por Meta que se utiliza principalmente para crear interfaces de usuario interactivas y eficientes. React utiliza una estructura de árbol conocida como Virtual DOM, la cual ayuda a la optimización de imágenes por segundo (fps), permitiendo así que las aplicaciones sean rápidas y respondan a las acciones del usuario. De acuerdo con Avalos y Guaillas [16], React.js se puede utilizar junto con varias tecnologías y herramientas como pueden ser: JSX (extensión de sintaxis HTML), CSS, Webpack, Babel

(convertidor de formato para navegadores), Redux (biblioteca de gestión) y React Router (gestión de navegación).

2.5.2 Visual Studio Code

Editor de código fuente conocido por ser ligero y de código abierto, desarrollado por Microsoft y enfocado principalmente para programadores y desarrolladores web. Visual Code ofrece potentes funcionalidades como el autocompletado del código, un sistema de depuración, un control de versiones integrado (git) y una gran cantidad de extensiones que facilitan el desarrollo al adaptarse a casi cualquier lenguaje o tecnología. Conforme con Zuñiga [17], este editor de código tiene un suporte multiplataforma permitiendo trabajar tanto en Windows, macOs y Linux. Por otra parte, puede presentar un gran consumo de recursos dando como resultado tiempo de carga lenta.

2.5.3 Node,js

Entorno de ejecución de JavaScript de código abierto y multiplataforma que permite a los desarrolladores construir aplicaciones en el lado del servidor y herramientas de red utilizando JavaScript. Basado en el motor V8 de Google Chrome, Node.js ejecuta código JavaScript fuera del navegador, facilitando la creación de aplicaciones escalables y eficientes. En la Tabla 1 se presenta un resumen de las principales características de Node,js.

Tabla 1: Principales características de Node.js.

Aspecto	Descripción
Características	- Basado en un modelo asíncrono y no bloqueante. - Utiliza un solo hilo con un modelo de eventos (Event Loop). - Permite crear aplicaciones altamente escalables a través de su naturaleza asincrónica.
Ventajas	- Alto rendimiento debido a su ejecución basada en el motor V8 y su modelo asíncrono. - Su fácil desarrollo permite utilizar JavaScript tanto en el frontend como en el backend. - Ideal para aplicaciones de chat, streaming y APIs en tiempo real.
Limitaciones	- No es adecuado para tareas pesadas en CPU debido a su arquitectura monohilo. - Su consumo de memoria es mayor debido a su naturaleza asíncrona. - Node.js es no es recomendable para cálculos pesados.

Fuente: Adaptado de [18]

2.6 Base de datos del sistema

2.6.1 SQLite

Biblioteca que proporciona un motor de base de datos SQL independiente. Esta base de datos es de origen público y puede utilizarse para cualquier propósito, ya sea de carácter comercial o privado, además, es un sistema de gestión de bases de datos relacional ligero, añadido en aplicaciones y muy utilizado para almacenar y gestionar datos de manera local. A diferencia de otros sistemas como MySQL o PostgreSQL, SQLite no requiere de un servidor independiente, dado que toda la base de datos se almacena en un solo archivo en el sistema de archivos del dispositivo, este punto lo hace ideal para aplicaciones móviles, dispositivos IoT y software que necesita capacidades de almacenamiento sin depender de una conexión a internet [19].

2.6.2 PostgreSQL

Es un sistema gestor de base de datos relacional (RDBMS), conocido por ser de código abierto, además de brindar gran seguridad, flexibilidad y robustez. Permite organizar, almacenar y consultar grandes volúmenes de datos utilizando el lenguaje SQL, permitiendo el soporte tanto de datos relaciones como objetos avanzados (JSON, XML, arrays, tipos personalizados). De acuerdo con Salcan [20], este gestor de base de datos cumple con las transacciones ACID (atomicidad, consistencia, aislamiento y durabilidad) desde hace más de dos décadas, además de integrar poderosos complementos como PostGIS para la gestión de información geoespacial.

2.7 Analítica de datos

Es el proceso de recolectar, transformar y analizar información para facilitar la toma de decisiones. Este enfoque permite a las organizaciones aprovechar al máximo sus datos para identificar patrones, predecir comportamientos y mejorar procesos. Conforme con Zapata [21], el análisis de datos es la utilización de la información con el fin de extraer conocimientos de una seria de datos, el objetivo principal de la analítica de datos es tomar mejores decisiones de acuerdo con la información obtenida.

Por otra parte, el manejo y análisis de grandes volúmenes de datos debe tener en cuenta tres características importantes, estas son: velocidad, volumen y variedad. Existen varios tipos de análisis de datos, los cuales se presentan en la Tabla 2.

Tabla 2: Tipos de análisis de datos

Tipos	Características
Descriptivo	- Resume datos históricos. - Identifica patrones y tendencias. - Utiliza reportes y dashboard. - Base para otros análisis.
Diagnóstico	- Detecta causas y relaciones. - Analiza anomalías y comportamientos. - Profundiza en los datos. - Explica sucesos pasados.
Predictivo	- Estima riesgos y tendencias. - Predice resultados futuros. - Utiliza machine learning. - Toma de decisiones anticipadas.
Prescriptivo	- Recomienda acciones óptimas. - Simula escenarios posibles. - Optimiza recursos y decisiones. - Se base en inteligencia artificial.

2.8 Metodología Mobile-D

Enfocada principalmente en el desarrollo de aplicaciones móviles, debido a que se basa en prácticas de programación extrema (XP) y Scrum, adaptándolas a los retos que involucra el

desarrollo móvil, como la necesidad de un buen rendimiento, pruebas en múltiples dispositivos y ciclos de vida reducidos de las aplicaciones. Además, estudios como los de Abrahamsson [22] et al. y Kaur [23] señalan que, si bien Mobile-D está diseñada especialmente para el desarrollo de aplicativos móviles, muchos de sus principios también pueden aplicarse a sistemas multiplataforma, ya que está fundamentada en prácticas de metodologías agiles como Scrum y XP. Esto es especialmente relevante si el desarrollo del sistema inicia con el apartado móvil como eje central, complementándolo posteriormente con el aplicativo web.

Conforme a lo anterior, se ha considerado importante mencionar el proyecto titulado "Implementación de un aplicativo móvil y web para la gestión administrativa de la empresa consta usando la metodología Mobile-D" [24]. Esto demuestra que dicha metodología ha sido aplicada de manera efectiva en el desarrollado de sistemas multiplataforma. Para destacar las características más importantes de las metodologías base de Mobile-D (Programación extrema y Scrum), se presenta la Tabla 3, que compara las metodologías mencionadas anteriormente.

Tabla 3: Cuadro comparativo de metodologías.

	Programación Extrema (XP)	**SCRUM**	**Mobile-D**
Fases	- Planeación. - Diseño. - Desarrollo. - Pruebas.	- Reunión de planificación de Sprints. - Scrum diario. - Desarrollo durante el Sprint. - Revisión del Sprint. - Retrospectiva del Sprint.	- Exploración. - Inicialización. - Producción. - Estabilización. - Pruebas del sistema.
Enfoque	Su objetivo principal son las necesidades del cliente, las cuales aseguran el éxito en el desarrollo del software.	Su enfoque se centra en los requisitos del cliente, para comenzar con el desarrollo en base a dichos requisitos.	Su objetivo es medir el nivel de satisfacción de los usuarios finales.
Programación	- Programación en grupo. - Jornadas largas de trabajo. - Retroalimentación mutua en base al código.	La puntuación de prioridad asignada a cada tarea determina el tiempo de programación.	La programación se realiza en grupo permitiendo mejorar la etapa de difusión de conocimiento dentro del grupo de trabajo.
Documentación	- Historias de usuario. - Tarjetas de clase, responsabilidades y colaboración (CRC).	- Product backlog. - Sprint backlog. - Burndown chart. - Definición de hecho. - Definición de culminación.	- StoryCards. - StoryBoards.
Principios orientados al desarrollo	Orientados a la gestión de proyectos.	Mejora del grupo de desarrolladores.	Orientado al desarrollo de aplicaciones móviles.

Fuente: Adaptado de [25].

Por otra parte, su estructura se divide en cinco fases claves que garantiza una implementación eficiente como se muestra en la Figura 3.

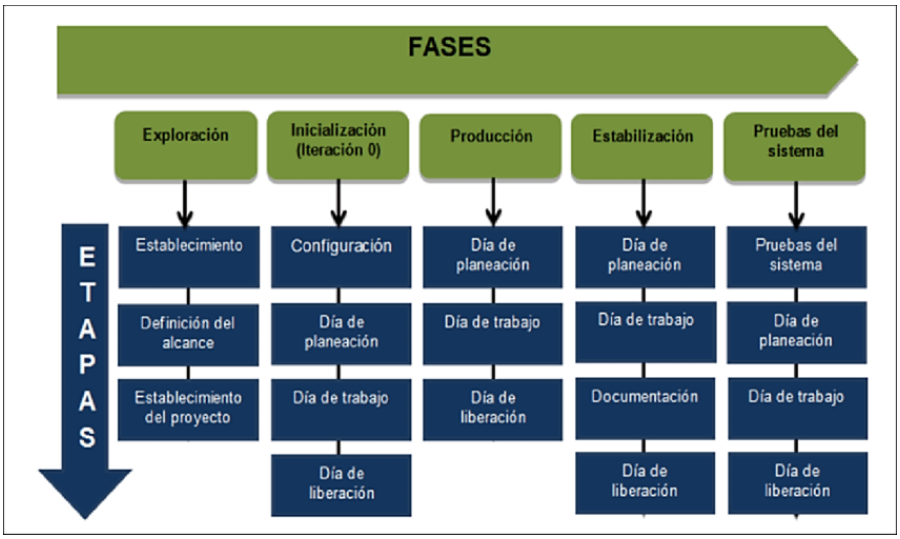


Figura 3: Ciclo de Mobile-D.
Fuente: [26]

2.8.1 Fases de la metodología Mobile-D

Fase 1: Exploración

En esta fase inicial se definen los objetivos del proyecto, al analizar los requerimientos del cliente y realizar estudios de factibilidad, además, se identifican riesgos técnicos y se elabora un plan de trabajo basado en iteraciones cortas. También se definen los roles de cada uno de los desarrolladores y finalmente se crea una versión beta del backlog de la aplicación. Según Sepa [27], esta etapa tiene como objetivo formar las bases del proyecto, por otra parte, funciona como un indicador de las expectativas respecto a la aplicación móvil. Igualmente, el autor menciona que los productos resultantes de esta fase son: requisitos iniciales, plan de proyecto, descripción de los procesos y plan de capacitación.

Fase 2: Inicialización

Durante esta etapa, se realiza el diseño de la arquitectura y se establecen los criterios de calidad del software, asimismo, se eligen las herramientas de desarrollo y los frameworks adecuados para la aplicación. Minina [28] menciona, que en esta fase se preparan e identifican los recursos que se crean necesarios. La fase de iniciación se divide en cuatro sub-fases que son: puesta en marcha del proyecto, planificación, día de prueba y finalmente salida.

Fase 3: Producción

Esta es la fase en la que se desarrolla el software de manera incremental e iterativa. Las funcionalidades se implementan siguiendo principios agiles como las pruebas frecuentes y la integración continua, igualmente, se deben realizar reuniones diarias que permitan evaluar los avances y analizar los problemas detectados. Conforme con Sepa [27], en esta fase se

incluye la implementación real mediante la aplicación del ciclo de desarrollo iterativo e incremental como se muestra en la Figura 4.

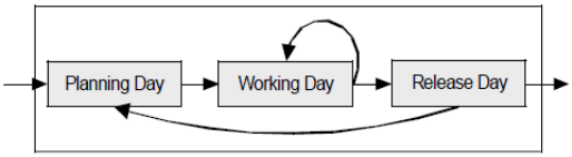


Figura 4: Proceso de la fase de producción.
Fuente: [6].

- Días de planificación (Planning Day): Su objetivo es definir el contenido a través de iteraciones y la aceptación de los usuarios, que se serán usados el día de la presentación final del proyecto.
- Días de trabajo (Working Day): En esta etapa se desarrollan las funciones del software de manera controlada y gestionada.
- Días de lanzamiento (Release Day): Se valida y verifica la funcionalidad del software para comprobar la aceptación del usuario.

Al culminar esta fase se obtienen las anotaciones de desarrollo, funcionalidades en producción, esquema de interfaz de usuario, historias de usuario y por último los requisitos que fueron modificados.

Fase 4: Estabilización

Aquí se realizan las pruebas que ayudan a detectar errores y mejorar el rendimiento del software. Para lograr esto se aplican pruebas automatizadas y manuales que garantizaran la estabilidad y usabilidad del aplicativo. Además, en esta etapa se optimizan apartados como la eficiencia del código, el consumo de recursos y la experiencia final del usuario, basado en el proceso que se puede observar en la Figura *5*.

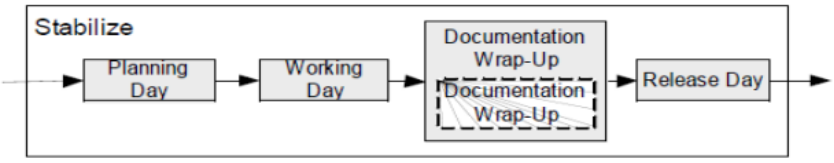


Figura 5: Proceso de la fase de estabilización.
Fuente: [6]**.**

Fase 5: Pruebas del sistema

Según Minina [28], el propósito de esta fase es que el aplicativo sea estable y funcional para los usuarios finales. La aplicación terminada se integrará y se le realizaran pruebas en base a los requerimientos del cliente, permitiendo eliminar todos los errores encontrados. Cabe recalcar que esta fase se divide en dos etapas, específicamente la de pruebas del sistema y la de corrección de errores, como se puede observar más detalladamente en la Figura 6.

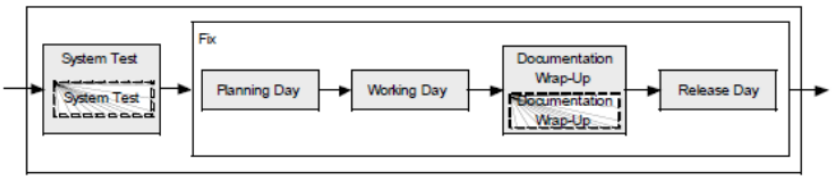


Figura 6: Proceso de la fase de pruebas del sistema.
Fuente: [6].

- Pruebas del sistema (System Test): En esta etapa se prueba el sistema como se define en el modelo de tarea de prueba, donde los defectos encontrados se trabajan en la iteración de corrección de errores (Fix).
- Corrección de errores (Fix): Este apartado es una variación de la iteración normal, sin embargo, ninguna funcionalidad nueva es añadida. El motivo para esta iteración son los defectos encontrados en la fase de pruebas.

2.9 Norma ISO/IEC 25012:2008

Como parte de la familia de normas SQuaRE (Software Product Quality Requirements and Evaluation), establece un modelo integral para evaluar la calidad de los datos almacenados en sistemas informáticos. Este modelo define quince características esenciales para garantizar que los datos sean adecuados, confiables y útiles, organizándolas en dos perspectivas principales. La calidad de datos evalúa propiedades fundamentales, como la exactitud, completitud, consistencia y credibilidad, asegurando que los datos reflejen fielmente la realidad que representan, además de otras características que se muestran en la Figura 7.

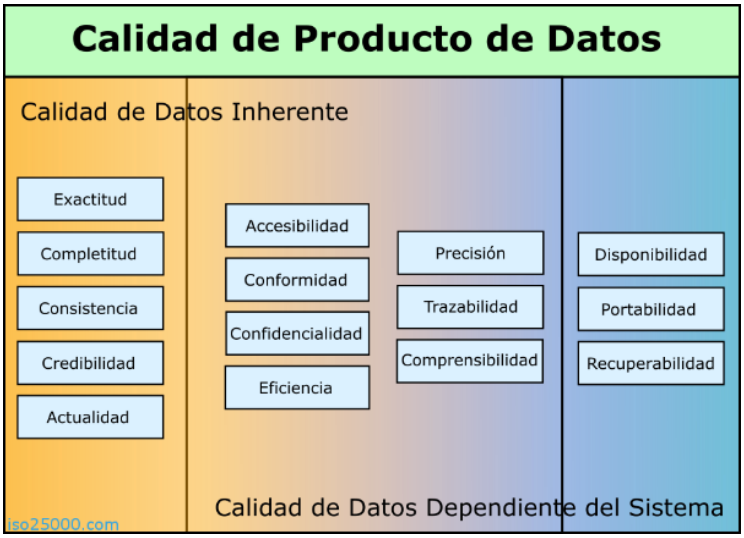


Figura 7: Características de la norma ISO/IEC 25012:2008.
Fuente: [29]

2.9.1 Fiabilidad

Hace referencia a el grado de confianza que se tiene en su precisión, consistencia y fiel representación de la realidad. En este sentido, la fiabilidad es una propiedad especifica la norma ISO/IEC 25012:2008 establece un marco conceptual en el que la fiabilidad se puede

evaluar mediante características dependientes e inherentes del sistema. Según Gualo et al. [30], se recomienda el uso del modelo de calidad de datos definido en la norma ISO/IEC 25012:2008, que establece características, propiedades y medidas concretas para evaluar la fiabilidad a través de los indicadores implícitos en esta norma. Dentro de estas se escogieron tres indicadores clave, específicamente la eficiencia, precisión y disponibilidad, las cuales se clasifican en base al tipo de dato, como se observa en la Tabla 4.

Tabla 4: Clasificación de características seleccionadas de la norma ISO 25012:2008

Características	Inherente	Dependiente del sistema
Eficiencia	X	
Precisión	X	
Disponibilidad		X

Fuente: Adaptado de [31].

Estudios recientes evidencian que estos tres indicadores son dimensiones mediante las cuales se puede evaluar la fiabilidad de datos. El artículo "Overview of Data Quality: Examining the Dimensions, Antecedents, and Impacts of Data Quality" [32], señala que estudios fundamentales en el área incluyen de forma habitual atributos como accuracy (precisión), timelines (eficiencia y acceso oportuno) y availability (disponibilidad) dentro de las dimensiones esenciales para valorar la calidad y fiabilidad (reliability) de la información.

- Eficiencia: Mide el rendimiento en el que los datos tienen atributos que pueden ser procesados en un contexto específico.
- Precisión: Analiza los atributos de los datos para que estos sean exactos y no cuenten con cambios en un contexto específico.
- Disponibilidad: Grado en el que los datos pueden ser obtenidos por usuarios o aplicaciones fácilmente en un contexto específico.

CAPÍTULO III. METODOLOGÍA

3.1 Tipo de Investigación

3.1.1 Según el objeto de estudio

La investigación se clasificó como **aplicada**, pues buscó emplear conocimientos teóricos y prácticos en el desarrollo de un sistema informático para la gestión de censos de salud del cantón Chambo.

Así también, se clasificó como una investigación **descriptiva**, debido a que se enfocó en detallar las características del sistema en relación con la fiabilidad de los datos, conforme a los criterios establecidos dentro de la norma ISO/IEC 25012:2008.

3.1.2 Según el tipo de variable

La presente investigación demostró un enfoque cuantitativo, dado que se basó en el análisis de datos numéricos obtenidos a partir de diversas simulaciones, que comprobaron como la aplicación manejó la entrada y persistencia de los datos, mediante indicadores específicos que evaluaron la eficiencia, precisión y disponibilidad de la información, conforme con la norma ISO mencionada anteriormente.

3.2 Diseño de la investigación

Esta investigación tomó un diseño no experimental dado que no se manipuló las variables de manera intencional, en este caso, se realizó y analizó las simulaciones aplicadas en el producto final en su contexto natural.

3.3 Población de estudio y tamaño muestra

La población objeto del presente estudio se consideró infinita, dado que se utilizaron herramientas de software con una carga de concurrencia simulada. En este sentido, no se estableció un tamaño de muestra, ya que no se seleccionó un subconjunto de individuos, sino que se trabajó con un número configurable de UV (usuarios virtuales) predefinidos en cada simulación, los mismos que sirvieron para representar de manera artificial el comportamiento de personas reales dentro del sistema. El análisis se centró en la evaluación cuantitativa de la fiabilidad de los datos del sistema multiplataforma, considerando indicadores como la eficiencia, precisión y la disponibilidad de la información, conforme a la norma ISO/IEC 25012:2008.

3.4 Técnicas de recolección de datos

Para la recolección de datos, se utilizó como técnica la experimentación mediante simulación, con el fin de obtener información precisa sobre los indicadores de la fiabilidad de los datos relacionados con el sistema de gestión de censo de salud. Se evaluó como la aplicación maneja la entrada y persistencia de los datos bajo distintas condiciones de carga, diseñándose escenarios controlados con situaciones reales de uso. Para ello, se usó herramientas especializadas de simulación de carga y rendimiento, como Apache Jmeter, K6

y bibliotecas de Python, las cuales permitieron medir parámetro claves establecidos en la fase de pruebas del sistema, además de brindar la posibilidad de generar la cantidad de usuarios virtuales adecuados para cada simulación.

3.5 Métodos de análisis y procesamiento de datos

Para evaluar el manejo de la fiabilidad de los datos del sistema multiplataforma, se llevó a cabo simulaciones de carga con Apache JMeter para la eficiencia, validaciones cruzadas entre los datos ingresados y almacenados mediante bibliotecas de Python para determinar la precisión, y pruebas de sincronización con K6 para analizar la disponibilidad. Cada una de estas simulaciones tienen un conjunto de indicadores técnicos detallados dentro de la metodología de desarrollo, permitiendo un análisis cuantificable del sistema bajo condicione controladas.

3.6 Identificación de variables

- **Variable dependiente**

Fiabilidad de los datos.

- **Variable independiente**

Sistema de gestión de censo.

3.7 Operacionalización de variables

Se presenta la Tabla 5 que contiene la operacionalización de las variables.

Tabla 5: Operacionalización de las variables.

Problema	Tema	Objetivos	Variables	Conceptualización	Dimensión	Indicadores
¿Cómo impactará la implementación de un sistema para la gestión del censo de salud en la fiabilidad de los datos del Centro de Salud Chambo, considerando los criterios de eficiencia, precisión y disponibilidad establecidos por la norma ISO/IEC 25012:2008?	Sistema para la gestión de censo de salud en el Centro de Salud Chambo utilizando el framework Flu tter.	**General** Implementar un sistema para la gestión de censo de salud en el Centro de Salud Chambo utilizando el framework Flutter. **Específicos** • Investigar la metodología de desarrollo ágil Mobile-D, para la implementación de un sistema que integre componentes de datos enfocados en el Centro de Salud Chambo. • Desarrollar una aplicación web y móvil para la recopilación y análisis de información de censo en el centro de salud Chambo. • Evaluar la fiabilidad de los datos del sistema de gestión de censo de salud conforme con los criterios establecidos por la norma ISO/IEC 25012:2008.	**Independiente** Sistema de gestión de censo. **Dependiente** Fiabilidad de los datos.	Un sistema de información (SI) es definido como un conjunto integrado de componentes que interactúan entre sí para recopilar, procesar, almacenar y distribuir información, con el fin de apoyar actividades de toma de decisiones, coordinación, análisis y visualización en organizaciones. Según la ISO/IEC 25012:2008, la fiabilidad de los datos se considera una característica esencial dentro del modelo de calidad de datos, debido a que garantiza que estos sean consistentes, precisos y estén accesibles cuando se necesiten. Esto asegura su utilidad en procesos de toma de decisiones, minimizando riesgos asociados a errores o incoherencias en la información.	Desarrollo de software. Calidad de datos.	Independiente. • Módulos. • Tamaño de la aplicación. • Compatibilidad del sistema. • Tiempo de desarrollo. Dependiente. • Eficiencia de datos. • Precisión de los datos. • Disponibilidad.

3.8 Metodología de desarrollo (Mobile-D)

Fase 1: Exploración

Esta fase determinó el punto de partida para el desarrollo del aplicativo, abarcando los requisitos iniciales para el sistema móvil y web, la identificación de los grupos de interés, la evaluación de los requisitos funcionales y no funcionales, así como la delimitación del alcance del proyecto.

- **Requisitos iniciales**

Como parte del levantamiento de requisitos iniciales para el desarrollo del sistema, se realizó una visita de campo al centro de salud Chambo. En la fase inicial se definieron los requisitos funcionales y no funcionales del sistema, orientados a mejorar y optimizar el levantamiento de información del área de la salud en escuelas del cantón Chambo. Como eje principal del sistema, se estableció la necesidad de desarrollar un aplicativo móvil capaz de recolectar y almacenar la información con soporte para operaciones de manera offline, así como un aplicativo web que permita consumir los datos para su posterior visualización, análisis y generación de reportes mediante analítica de datos.

- **Identificación de los grupos de interés**

Las personas identificadas que serán parte del desarrollo de la aplicación se muestran en la Tabla 6.

Tabla 6: Grupos de interés.

Grupo	Descripción
Desarrolladores	Grupo de estudiantes de la Universidad Nacional de Chimborazo, encargados del desarrollo del sistema móvil y web.
Encuestadores	Personal médico zonal encargado del manejo de la aplicación móvil.
Encuestados (alumnos)	Grupo de interés prioritario, que podrán contar con el monitoreo de salud, una vez aplicada la encuesta.
Encargado del Centro de Salud	Es la persona encargada de administrar y consumir la información del sistema móvil y web.

- **Requisitos funcionales**

En la Tabla 7 se presentan los requisitos funcionales del proyecto.

Tabla 7: Requisitos Funcionales.

Id	Requerimiento	Descripción	Prioridad
RF-1	Autenticación de usuario	El encuestador al iniciar sesión en el aplicativo móvil accederá a una pantalla con validación de credenciales y rol de usuario.	Alta

RF-2	Descarga de datos (estudiantes, unidades y campañas de vacunación)	El sistema descarga de forma local los datos precargados desde el servidor sobre unidades educativas, estudiantes y campañas de vacunación.	Alta
RF-3	Comprobación de la unidad educativa	Tras la descarga, el encuestador puede verificar la información de la unidad educativa antes de registrarla.	Media
RF-4	Comprobación de la información del estudiante	La validación de estudiantes se realiza mediante el número de cedula presente en la lista precargada.	Media
RF-5	Estado del registro de la encuesta	La interfaz presenta un módulo para gestionar el estado de las encuestas: completas, incompletas y pendientes.	Media
RF-6	Activación del módulo de encuestas	Una vez activado el estado de la encuesta, el encuestador puede registrar una encuesta vinculada a cada estudiante.	Alta
RF-7	Módulo de alimentación y nutrición	El sistema calcula el IMC y clasifica el estado nutricional del estudiante a partir de su talla y peso.	Alta
RF-8	Módulo de vacunación	El encuestador podrá registrar vacunas en función de la edad del estudiante.	Alta
RF-9	Módulo de campañas de vacunación	Atreves del módulo correspondiente, se registran las campañas de vacunación disponibles para el estudiante.	Alta
RF-10	Módulo de tamizaje visual	Se almacena la información relacionada con la salud visual del estudiante.	Alta
RF-11	Módulo de salud oral	El módulo odontológico permite registrar el estado de las piezas dentales del estudiante.	Alta
RF-12	Módulo de salud mental	Se genera un registro de la evaluación psicológica y emocional del estudiante.	Alta
RF-13	Módulo de higiene y saneamiento	Este módulo permite registrar las condiciones sanitarias básicas del estudiante.	Alta
RF-14	Sincronización de datos	Al completar los módulos, los datos almacenados localmente se sincronizan con el servidor cuando haya conexión a internet.	Alta
RF-15	Gestión de la sección offline	El aplicativo mantiene activa la sesión del encuestador cuando no tenga conexión a internet.	Alta
RF-16	Visualización de reportes	El sistema presenta gráficos y resúmenes estadísticos de la información recolectada en las encuestas.	Alta
RF-17	Módulo de registro de usuarios	El módulo de administración incluye la opción de registrar nuevos encuestadores.	Media

- **Requisitos no funcionales**

En la Tabla 8 se presentan los requisitos no funcionales del proyecto, que especifican los atributos y características del sistema.

Tabla 8: Requisitos No Funcionales.

Id	Requerimiento	Descripción
RNF-1	Lenguaje de programación	La aplicación móvil fue desarrollada utilizando Flutter, mientras que la aplicación web fue implementada con React.js.
RNF-2	Plataformas	La aplicación móvil se desarrolló para su uso en dispositivos Android, mientras que la visualización de reportes se gestionó en la aplicación web.

RNF-3	Interfaz	El sistema móvil y web tendrá una interfaz intuitiva para el fácil manejo del usuario.
RNF-4	Topologías de bases de datos	Las bases de datos utilizadas en el sistema son, SQLite y PostgreSQL, las cuales se encargaron del almacenamiento de datos y la implementación de la metodología offline.
RNF-5	Fiabilidad de los datos	Los datos recolectados y enviados al servidor deben transmitirse de manera segura y confiable.

- **Definición del alcance**

En esta sección se detallan que aspectos serán considerados dentro del proyecto, definiendo así las limitaciones y el establecimiento del proyecto.

a) Limitaciones

Las limitaciones que tendrá el sistema para la gestión de censo se presentan en la Tabla 9.

Tabla 9: Limitaciones del desarrollo del proyecto.

Aspecto	Limitaciones
Aplicación Móvil	• Se podrá generar una carga masiva de datos del estudiante, unidades educativas, alimentación, vacunación, salud oral, tamizaje visual, salud mental, higiene y saneamiento. • El aplicativo se ejecutará en dispositivos Android, permitiendo la sincronización de la base de datos local con el servidor. • Se implementarán roles de usuario para el inicio de sesión. • Permitirá la carga de datos en el dispositivo de manera offline. • Este apartado permitirá visualizar los datos recolectados mediante la aplicación móvil.
Aplicación Web	• Únicamente el personal asignado podrá visualizar los reportes detallados del censo aplicado a la población estudiantil. • Contará con roles de usuario, donde el administrador designará los permisos del personal médico.
Implementación Tecnología	• La base de datos central contará con una estructura que permitirá la carga masiva de datos. • Se permitirá la sincronización de la información de la aplicación móvil al servidor central, permitiendo la actualización continua de la información.
Módulos de Desarrollo	• Módulo de Alimentación y Nutrición. • Módulo de Vacunación. • Módulo de Tamizaje Visual. • Módulo de Salud Oral. • Módulo de Salud Mental. • Módulo de Higiene y Saneamiento.

Fase 2: Inicialización

El objetivo de esta fase es explicar las actividades de desarrollo para el diseño del aplicativo móvil y web. En esta etapa se definirán aspectos clave como las configuraciones del ambiente de desarrollo, la planificación de las fases basadas en iteraciones, el diseño de la aplicación web y móvil, el diagrama de la base de datos y de caso de uso y el esquema de navegabilidad.

- **Configuración del ambiente de desarrollo**

Este punto está dirigido exclusivamente para los desarrolladores del sistema y tiene como objetivo configurar un espacio de trabajo idóneo para el desarrollo del aplicativo, a continuación, la Tabla 10 describe las herramientas utilizadas paras la preparación del ambiente.

Tabla 10: Ambiente de desarrollo basado en el modelo MVC.

Elemento	Detalle
Tipo de proyecto	Aplicación hibrida
Vista de la aplicación móvil	Widgets de Flutter
Vista de la aplicación web	React.js
Controlador de la aplicación móvil	Dart
Controlador de la aplicación web	Node.js
Base de datos para la aplicación móvil	SQLite
Base de datos para la aplicación web	PostgreSQL
IDE de desarrollo para la aplicación móvil	Android Studio
IDE de desarrollo para la aplicación web	Visual Studio Code

- **Planificación de fases**

En esta etapa se divide el trabajo en fases secuenciales y ordenadas que se basan en iteraciones. Este proceso permite detallar las secuencias de las fases que guiaran el desarrollo tanto del aplicativo móvil y web. Siendo la Tabla 11 la responsable de describir cada una de las iteraciones.

Tabla 11: Planificación de las fases.

Fase	Iteración	Descripción
Exploración	Iteración 0 (semana 1)	Establecimiento de los requerimientos iniciales, grupos de interés, requisitos funcionales y no funciónales, además de las limitaciones.
Inicialización	Iteración 0 (semana 1)	Selección de las herramientas y tecnologías para el ambiente de desarrollo.
	Iteración 1 (semana 2)	Diseño de la aplicación, diseño del diagrama de la base de datos, esquema de navegabilidad y diagrama de caso de uso.
Producción	Iteración 2 (semana 3)	Implementación de la funcionalidad de la autenticación de usuarios, basado en un sistema de roles.
	Iteración 2 (semana 3)	Implementación del dashboard principal, con las funcionalidades de registro de unidades educativas, estudiantes y encuestas, además de la sincronización.
	Iteración 3 (semana 4)	Implementación de la funcionalidad que permite trabajar con el módulo de alimentación y nutrición.
	Iteración 4 (semana 5)	Implementación de la funcionalidad que permite trabajar con el módulo de vacunación.
	Iteración 5 (semana 6)	Implementación de la funcionalidad que permite trabajar con el módulo de tamizaje visual.
	Iteración 6 (semana 7)	Implementación de la funcionalidad que permite trabajar con el módulo de salud oral.
	Iteración 8 (semana 9)	Implementación de la funcionalidad que permite trabajar con el módulo de salud mental.
	Iteración 9 (semana 10)	Implementación de la funcionalidad que permite trabajar con el módulo de higiene y saneamiento.

	Iteración 10 (semana 11)	Implementación del login del apartado web.
	Iteración 10 (semana 11)	Implementación del dashboard principal de la página web que contara con los apartados de gestión de usuarios, consulta de información de cada estudiante, análisis y carga de datos.
	Iteración 11 (semana 12)	Implementación de la funcionalidad para la gestión de usuarios.
	Iteración 11 (semana 12)	Implementación de la funcionalidad para la consulta de información de cada estudiante.
	Iteración 12 (semana 13)	Implementación de la funcionalidad para el análisis de datos.
	Iteración 12 (semana 13)	Implementación de la funcionalidad para la carga de datos.
	Iteración 13 (semana 14)	Establecimiento de las interfaces finales del aplicativo móvil.
	Iteración 13 (semana 14)	Establecimiento de las interfaces finales del aplicativo web.
Estabilización	Iteración 14 (semana 15)	Implementación de sistemas de validación de caracteres y controles específicos.
	Iteración 14 (semana 15)	Corrección de cálculos de valores específicos de cada uno de los módulos.
Prueba del sistema	Iteración 15 (semana 16)	Realización de evaluaciones y pruebas del sistema, que permitirán analizar los resultados obtenidos del aplicativo.

- **Diseño del sistema multiplataforma**

Esta etapa permitió definir la estructura visual, funcional y técnica que tendrá el sistema, asegurando que se cumpla con los requerimientos iniciales establecidos para el apartado móvil y web. La Figura 8 muestra el diseño del aplicativo y sus componentes principales.

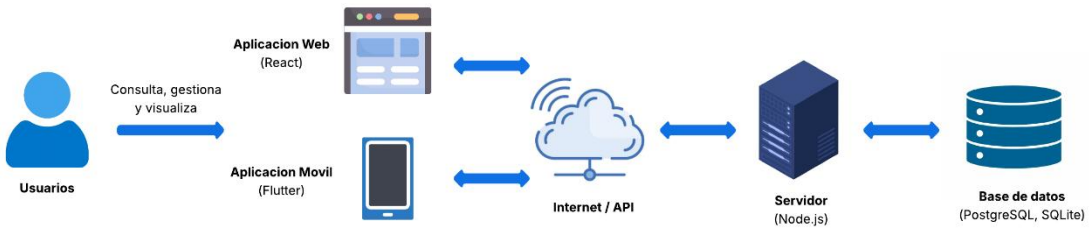


Figura 8: Diseño del sistema multiplataforma

- **Diagrama de la base de datos**

Para garantizar una correcta gestión de la información recolectada a través del aplicativo móvil y su posterior visualización mediante el aplicativo web, se diseñó un diagrama de base de base de datos estructurado que define la relación entre las entidades del sistema. En total se consideraron 13 tablas, cada una con atributos claves y sus distintos tipos de datos, que incluyen información relacionada con usuarios, encuestas, unidades educativas, estudiantes, campañas de vacunación y módulos específicos de salud (nutrición, salud oral, salud mental, tamizaje visual, entre otros). La Figura 9 presenta el modelo diseñado para este sistema, donde se evidencian las claves principales y foráneas que conectan los distintos módulos del censo de salud.

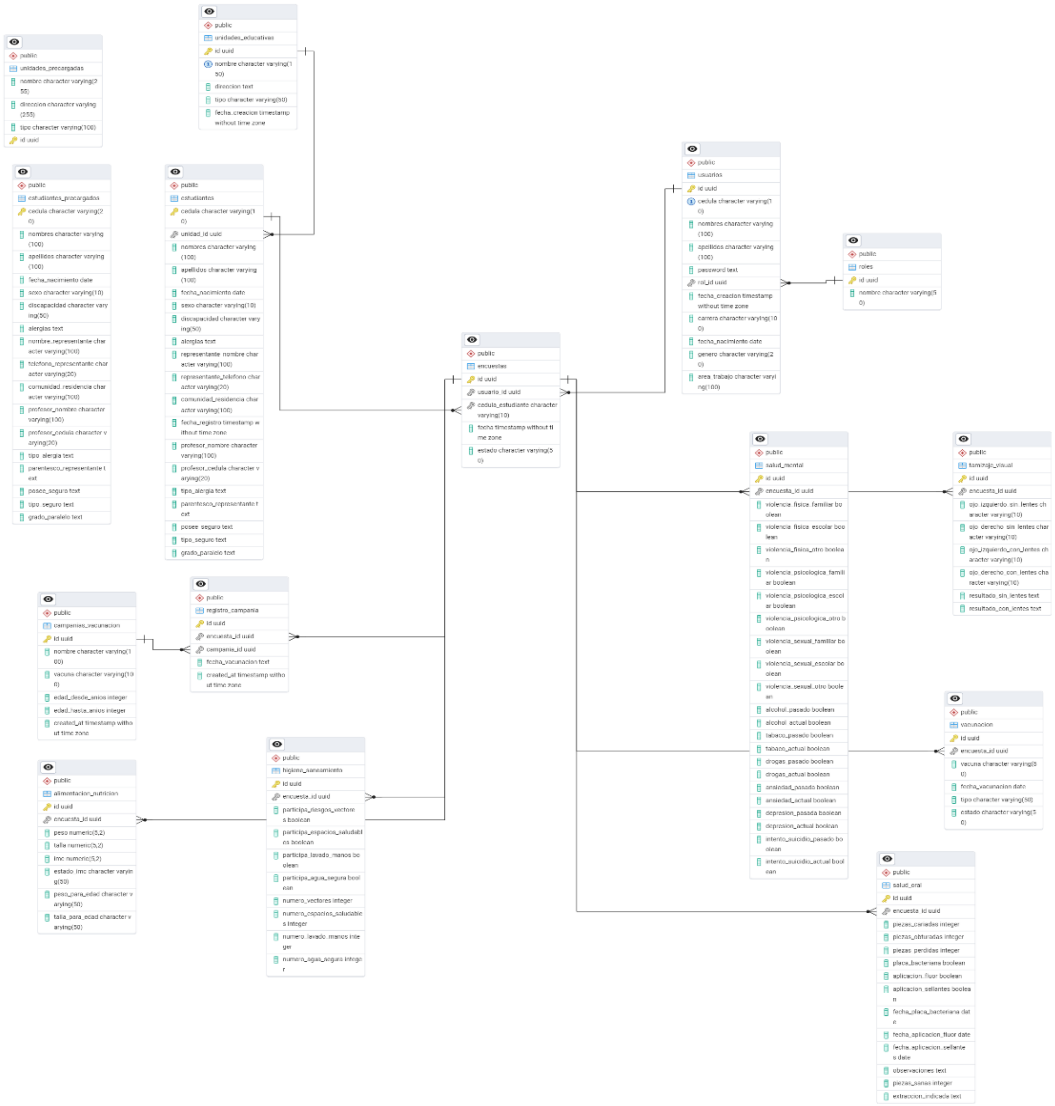


Figura 9: Diagrama de la base de datos.

Diccionario de datos de la base de datos

a) Tabla de usuarios

En la Tabla 12 se hace referencia a todos los campos que contiene la tabla usuarios. Para profundizar más sobre cada una de las tablas y sus características, visualizar el Anexo I.

Tabla 12: Diccionario de la tabla usuarios.

Campo	Tipo de Dato	Descripción
id	Uuid	Identificador único del usuario
cedula	varchar(10)	Cédula del usuario
nombres	varchar(100)	Nombres del usuario
apellidos	varchar(100)	Apellidos del usuario
password	Text	Contraseña encriptada
rol_id	Uuid	Rol del usuario (FK)
fecha_creacion	Timestamp	Fecha de creación

carrera	varchar(100)	Carrera del usuario
fecha_nacimiento	Date	Fecha de nacimiento
genero	varchar(20)	Género
area_trabajo	varchar(100)	Área de trabajo
estado	Varchar	Estado del usuario (Activo/Inactivo)

- **Esquema de navegabilidad**

Para definir de manera más clara el recorrido que realizaran los usuarios dentro de la aplicación móvil y web, se diseñó los esquemas de navegabilidad que representan las pantallas, módulos y secciones disponibles. Estos diagramas permiten visualizar el flujo lógico de navegación, identificando como el usuario transita y accede entre las distintas secciones del sistema. Para facilitar la comprensión de la arquitectura funcional y las rutas posibles, se elaboró un esquema para cada apartado.

a) Esquema de navegabilidad del apartado móvil

El esquema de navegabilidad del aplicativo móvil, presentado en la Figura 10, define la jerarquía de flujos y pantallas con las que interactuara el usuario. La estructura comienza con una pantalla de inicio o Splash y un módulo de autenticación (login) con validación de credenciales. Si las credenciales son correctas, el usuario accede al menú principal, que le permitirá dirigirse a los módulos de registro, tanto de unidades educativas, encuestas o estudiantes. Dentro del registro de encuestas se habilita el módulo de salud, que agrupa submódulos especializados: alimentación y nutrición, vacunación (con acceso al registro de campañas de vacunación), tamizaje visual, salud oral, salud mental e higiene y saneamiento.

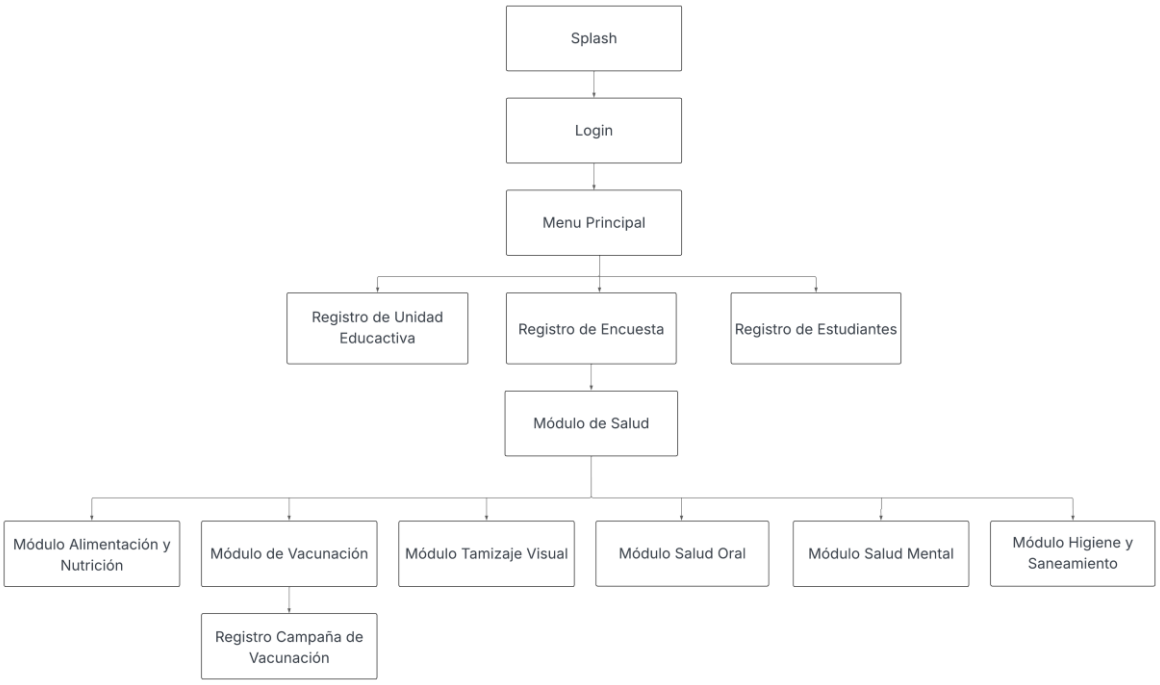


Figura 10: Esquema de navegabilidad de la parte móvil.

b) Esquema de navegabilidad del apartado web

De manera complementaria, se diseñó el esquema de navegabilidad para la aplicación web presentado en la Figura 11, enfocada principalmente en la consulta, gestión y análisis de la información. Al igual que el apartado móvil, el sitio web comienza con la validación de credenciales mediante la pantalla de login. Una vez autenticado, el usuario visualizara el menú principal que presenta las opciones de: gestión de usuarios (permitiendo el registro de nuevos usuarios y su visualización), visualizar estudiantes, analítica de datos, campañas de vacunación, avance de encuestas y carga de datos.

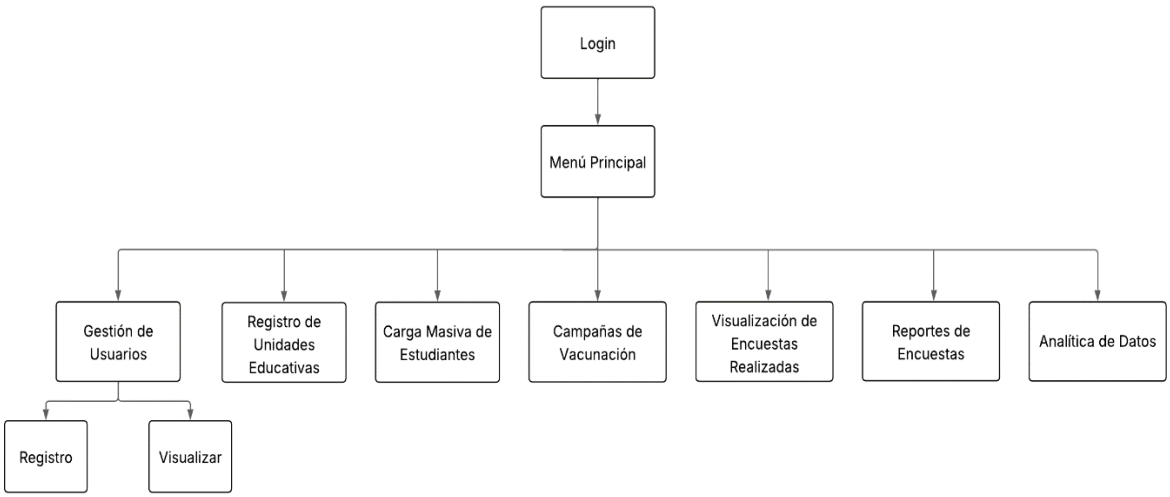


Figura 11: Esquema de navegabilidad de la parte web.

- **Diagrama de caso de uso**

a) Administrador

Las interacciones que realiza el administrador tanto del sistema móvil como web corresponden a las acciones y funcionalidades exclusivas a las que este perfil puede acceder dentro del sistema. Dichas interacciones se muestran en la Figura 12.

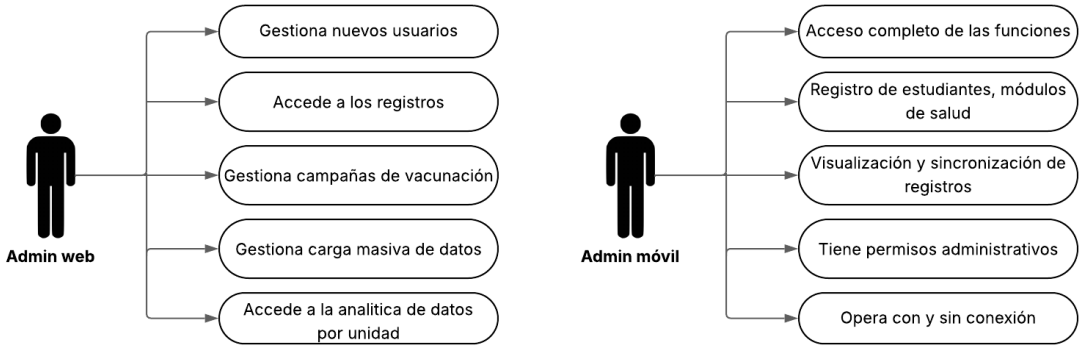


Figura 12: Diagrama de caso de uso para el administrador.

b) Encuestador

Por otro lado, las interacciones que realiza el encuestador responden al registro, almacenamiento y sincronización de datos de los diferentes módulos disponibles. Las funciones del encuestador se pueden observar más detalladamente en la Figura 13.

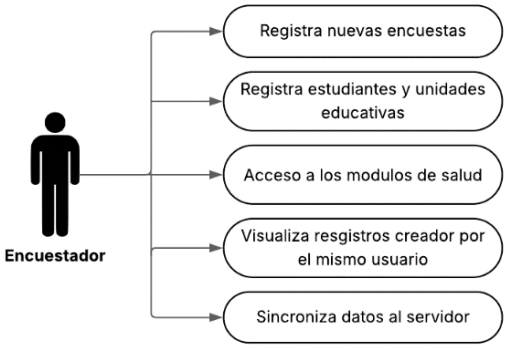


Figura 13: Diagrama de caso de uso para el encuestador.

c) Entidades adicionales

En cuanto a las entidades adicionales, se hace únicamente referencia a las funcionalidades complementarias a las cuales tiene acceso el rector en la interfaz web (Figura 14) y el odontólogo en el sistema móvil (Figura 15).

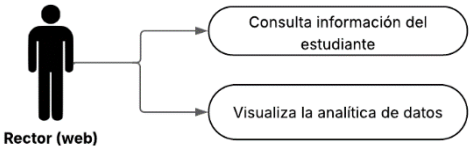


Figura 14: Diagrama de caso de uso para el rector.

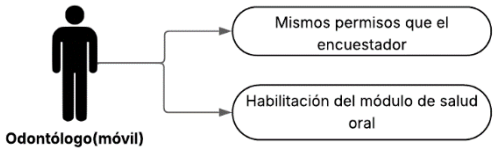


Figura 15: Diagrama de caso de uso para el odontólogo.

Fase 3: Producción y estabilización

En la fase de producción, correspondiente al desarrollo de la aplicación móvil y web, se describen todas las funcionalidades implementadas que en conjunto consolidan el trabajo realizado en las etapas anteriores, así mismo, en el apartado de la estabilización se integraron los distintos módulos del aplicativo.

a) Aplicación móvil

En la Figura 16 se visualiza la conexión y sincronización de la base de datos del aplicativo móvil con el backend. Para esta implementación, se desarrolló una API RESTful con Node.js que funciona como intermediaria para la sincronización de datos. La base de datos utilizada para la aplicación móvil fue SQLite, permitiendo almacenar los datos de manera local hasta que exista la conexión necesaria para subir la información al servidor.

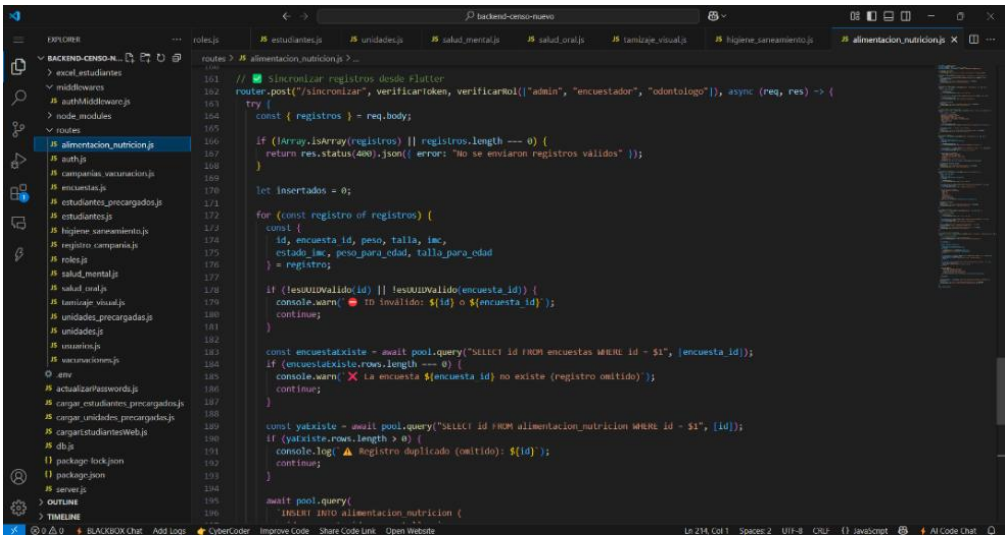


Figura 16: Conexión y sincronización de la aplicación móvil con el backend.

La Figura 17 muestra el módulo de inicio de sesión del aplicativo móvil. La autenticación se implementó mediante JSON Web Tokens (JWT). El proceso de validación por roles se desarrolló a través de un middleware en el backend. Para reforzar la seguridad, las contraseñas que se almacenaron en la base de datos fueron encriptadas con bcrypt.

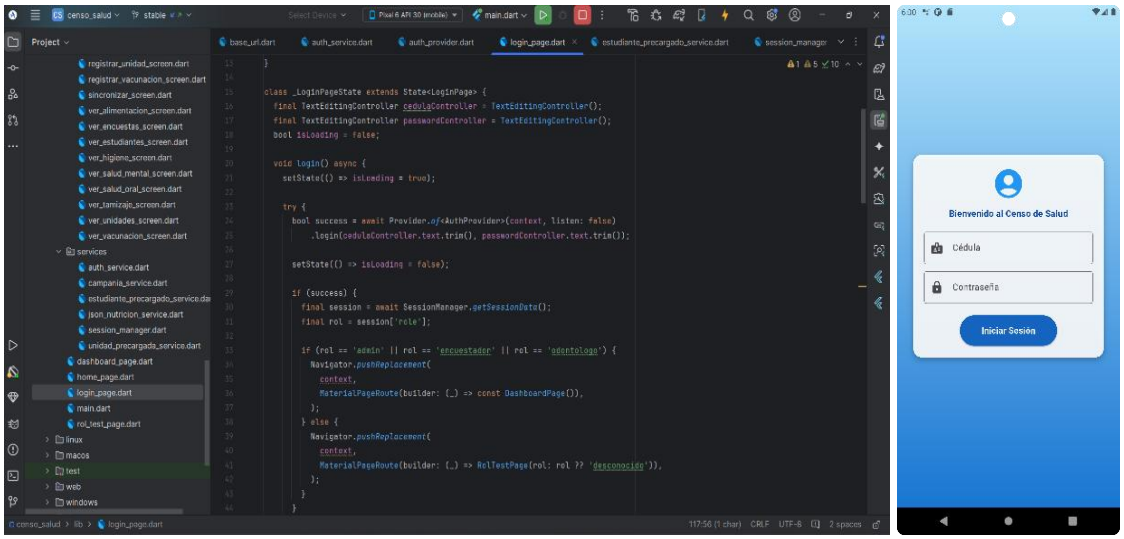


Figura 17: Login de la aplicación móvil.

En la Figura 18 se visualiza el menú principal, que incluye funciones generales de ingreso y visualización de unidades educativas y estudiantes, así como la sincronización completa de los datos. El desarrollo de la totalidad de la interfaz de usuario del aplicativo móvil fue empleado utilizando el framework Flutter. Las funcionalidades que incluye este layout son, el ingreso de datos y controladores de estado; la visualización de registros mediante consultas al backend, utilizando formatos de respuesta en JSON; la sincronización de la información entre el dispositivo móvil y la base de datos local, almacenando la información hasta que el usuario use la función de sincronización, momento en el que la información se envía al servidor.

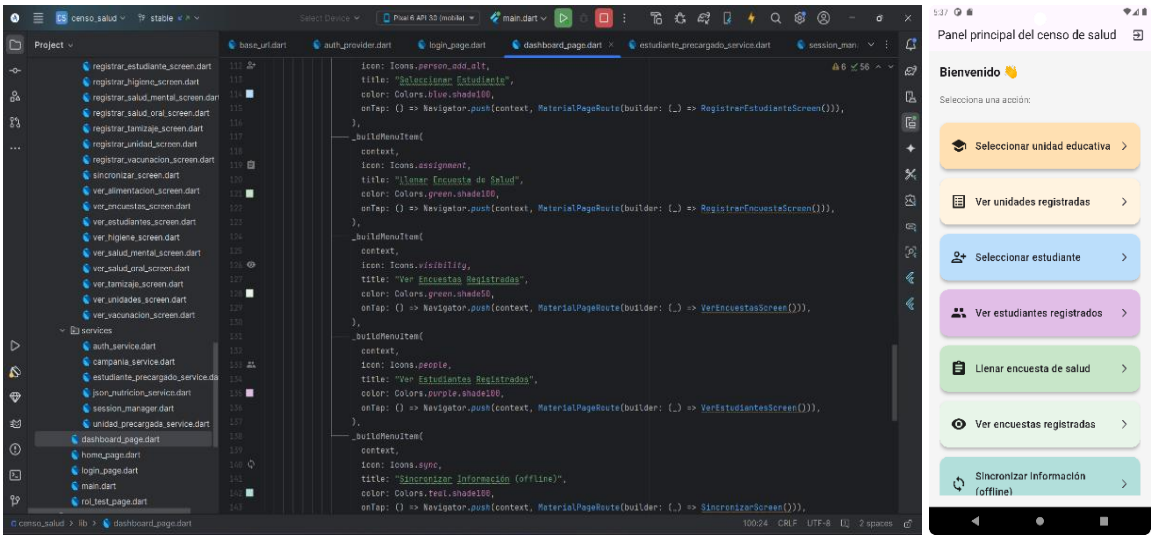


Figura 18: Menu principal de la aplicación móvil.

La Figura 19 muestra los apartados a los cuales se puede ingresar en función del rol del usuario, funcionalidad implementada a través de un sistema de autenticación y autorización basado en roles (RBAC, Role-Bassed Acces Control). Este mecanismo implementado mediante authMiddleware permitió establecer la lógica del sistema dividido en tres roles: administrador, encuestador y odontólogo. El desarrollo de esta funcionalidad intercepta cada petición al servidor y verifica la validez del token de usuario como los privilegios asociados al rol.

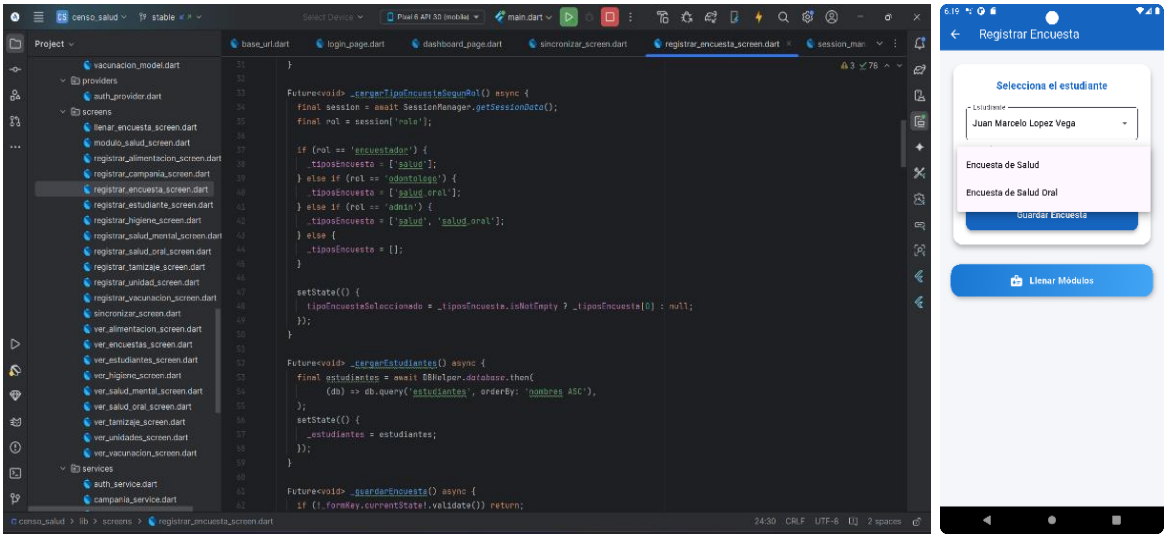


Figura 19: Registro de encuesta.

En la Figura 20 se presenta los distintos módulos de encuestas dependiendo del rol del usuario. De esta manera, el encuestador puede acceder a cinco módulos, mientras que, el odontólogo se limita al módulo de salud oral, permitiendo a ambos roles el registro y visualización de la información recolectada. A nivel técnico, se utilizó la funcionalidad de autenticación mediante una consulta a la base de datos, asimismo, se aplicaron principios de modularidad y reutilización de código al ser módulos similares.

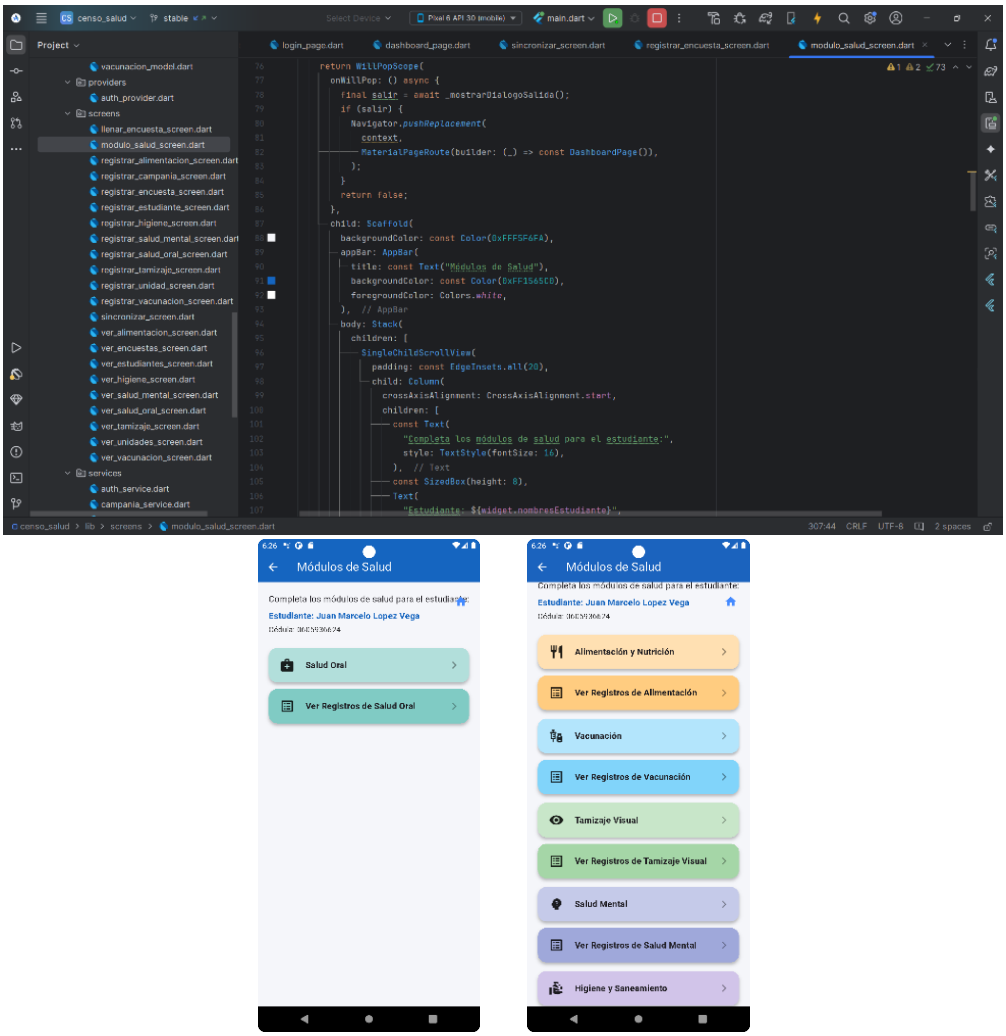


Figura 20: Menú principal de los módulos.

En la Figura 21 se puede observar un estatus de la sincronización de los datos, la misma que se activa una vez que el encuestador finaliza el registro de las encuestas. Durante la fase de recolección de información, los datos ingresados se almacenan temporalmente en la base de datos local. Posteriormente, mediante el botón sincronizar, toda la información se transfiere al servidor. El backend valida cada registro para evitar datos repetidos e inconsistentes, almacena la información y responde con un reporte de estado.

Cabe resaltar que el proceso de sincronización únicamente se ejecuta cuando el encuestador tiene conexión a internet; caso contrario, los datos permanecerán almacenados de manera local.

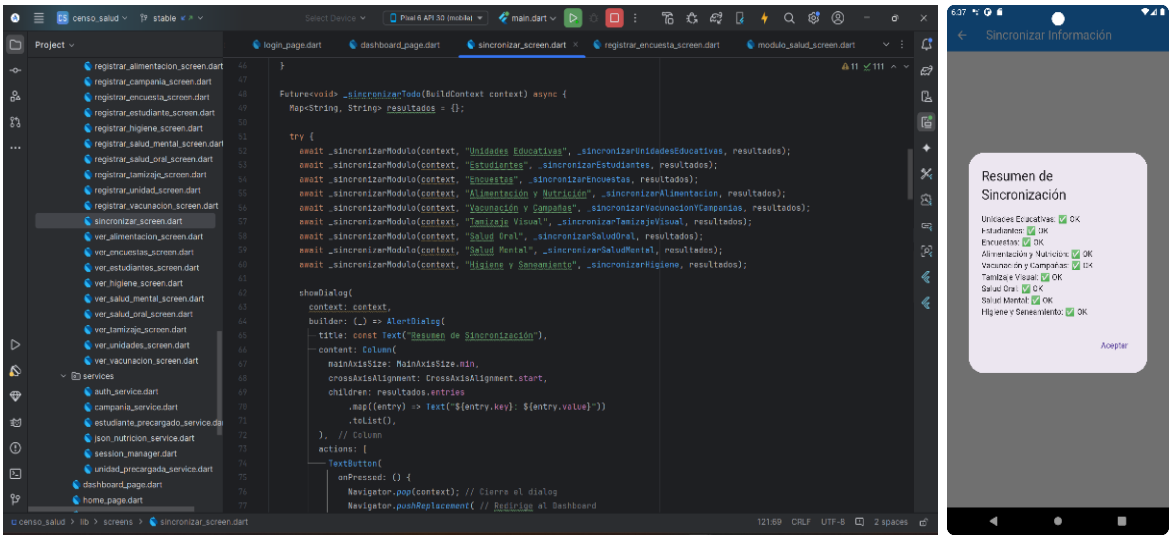


Figura 21: Sincronización de datos.

b) Aplicación web

La Figura 22 presenta la autenticación y validación por roles para el inicio de sesión. Este mecanismo mantiene la misma lógica de control que el aplicativo móvil, diferenciando los accesos según el perfil del usuario. Sin embargo, en este caso el backend se desarrolló mediante Node.js, mientras que la interfaz de usuario fue construida con React.

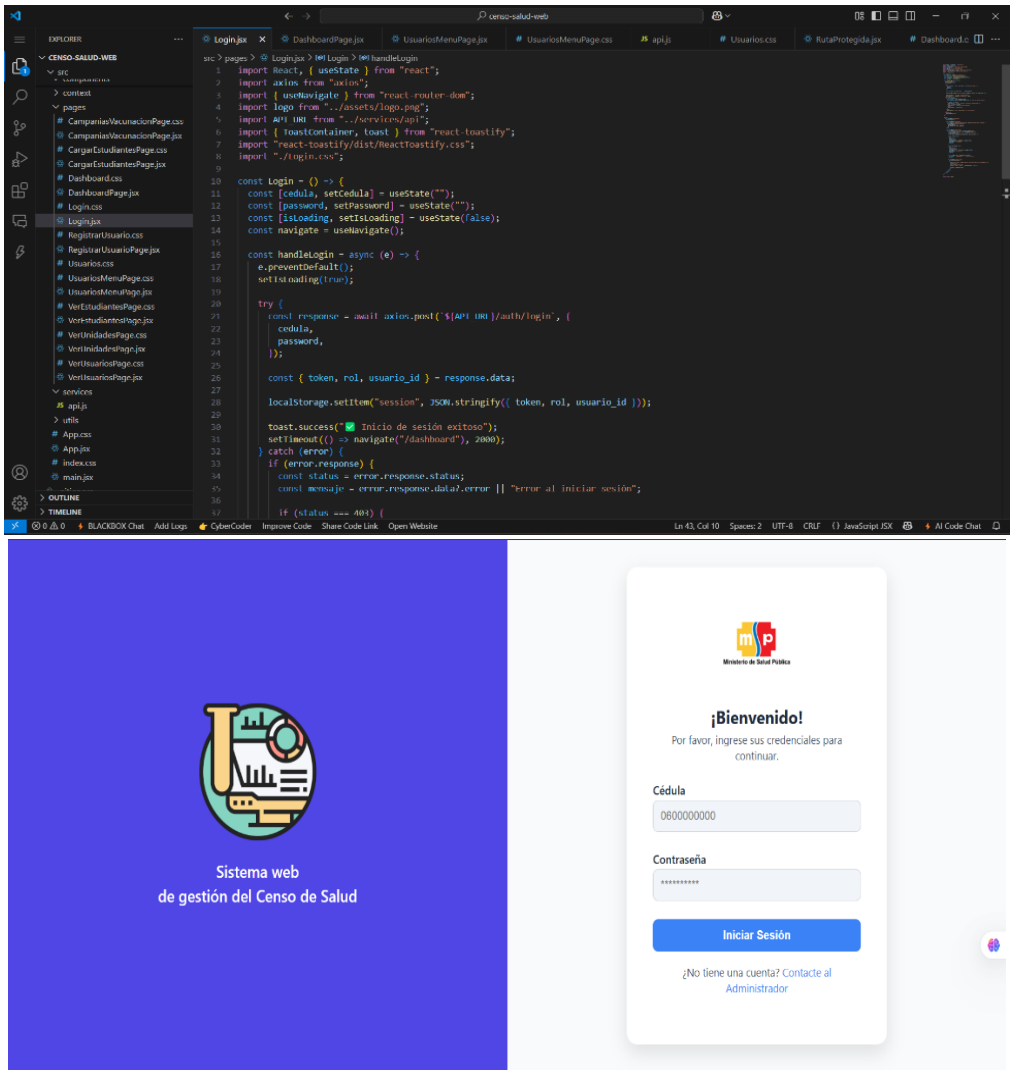


Figura 22: Login de la aplicación web.

En la Figura 23 se observa el menú principal del apartado web, permitiendo administrar las funciones del sistema mediante componentes modulares y servicios REST. Entre las funcionalidades principales, se encuentran los módulos: de gestión de usuarios, analítica de datos, campañas de vacunación, gestión de unidades educativas, carga masiva de datos en formato Excel, visualizar avance de encuestas y visualizar a los estudiantes registrados.

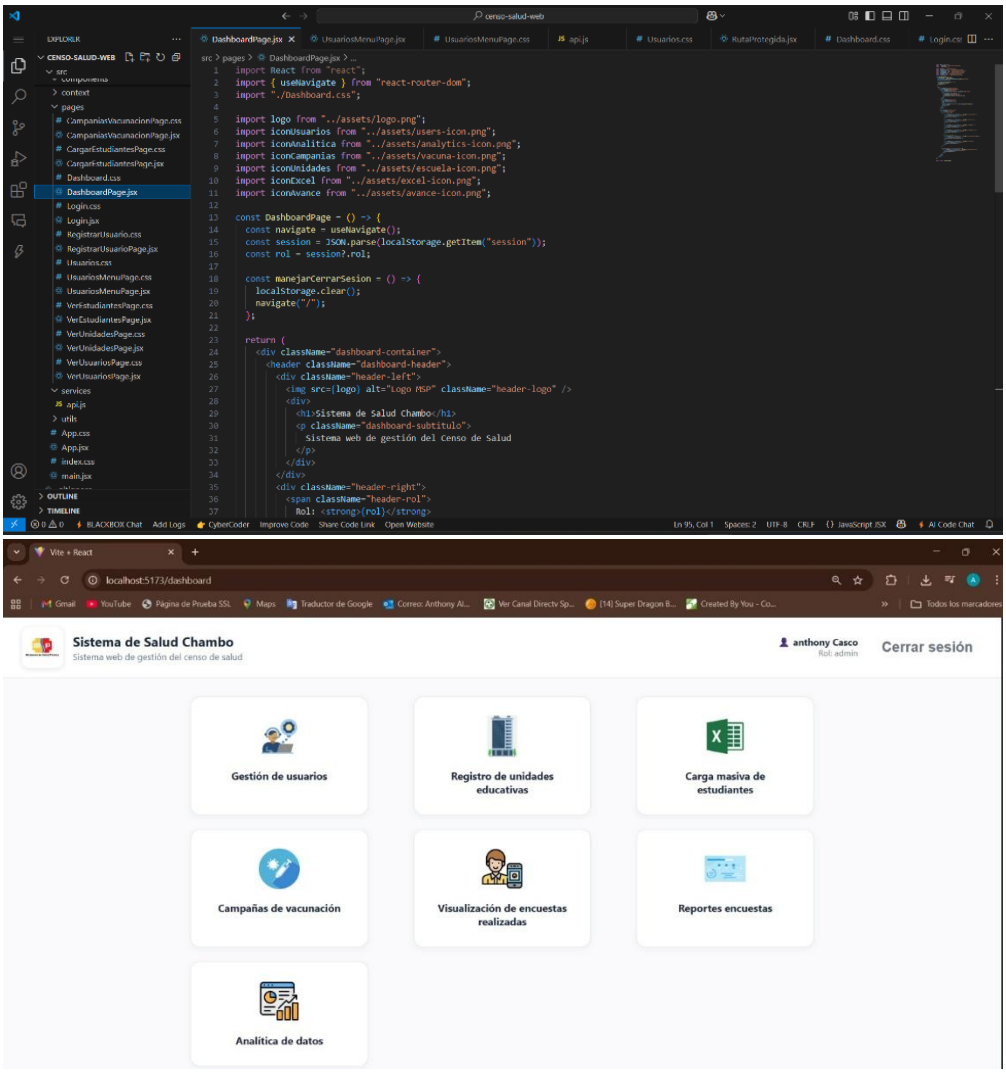


Figura 23: Menú del aplicativo web.

Los módulos presentados en la Figura 24, fueron desarrollados siguiendo un enfoque CRUD (créate, read, update, delete). Estos módulos se realizaron mediante un formulario controlado en el frontend, capturando los datos para posteriormente enviarlos al backend. Por otra parte, la visualización se implementó mediante una tabla dinámica. En cuanto al módulo de edición, el sistema permite actualizar la información en tiempo real. Finalmente, la búsqueda mediante filtros se implementó mediante consultas especificas en la base de datos.

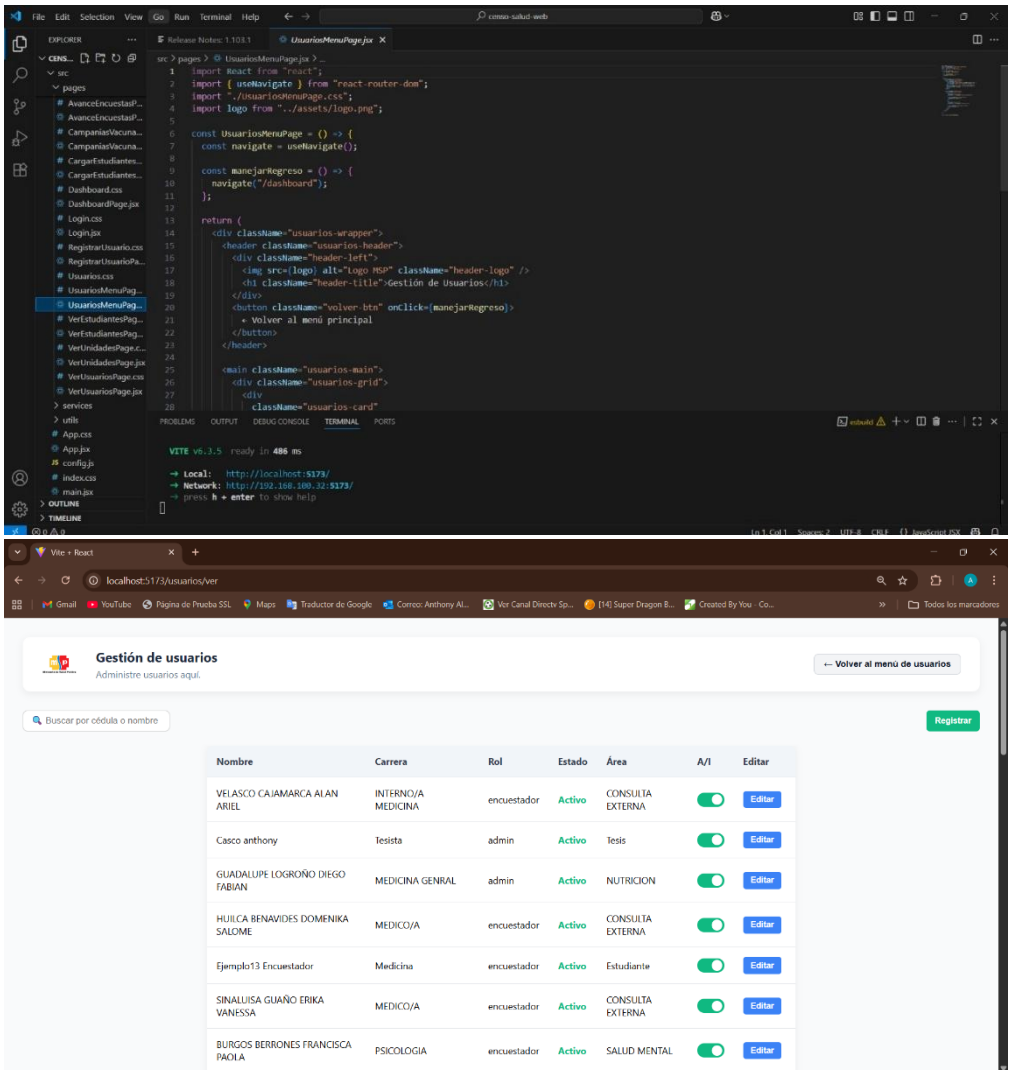


Figura 24: Módulos de gestión de usuario, campañas de vacunación, unidades educativas y visualización de estudiantes. (web).

La Figura 25 corresponde a la analítica de datos, cuyo diseñado fue un enfoque de la analítica descriptiva. La funcionalidad principal se basa en filtros dinámicos de búsqueda, permitiendo segmentar la información en base a distintos criterios. Los resultados se presentaron mediante tablas y gráficas interactivas, utilizando librerías especializadas como Chart.js para la visualización de datos.

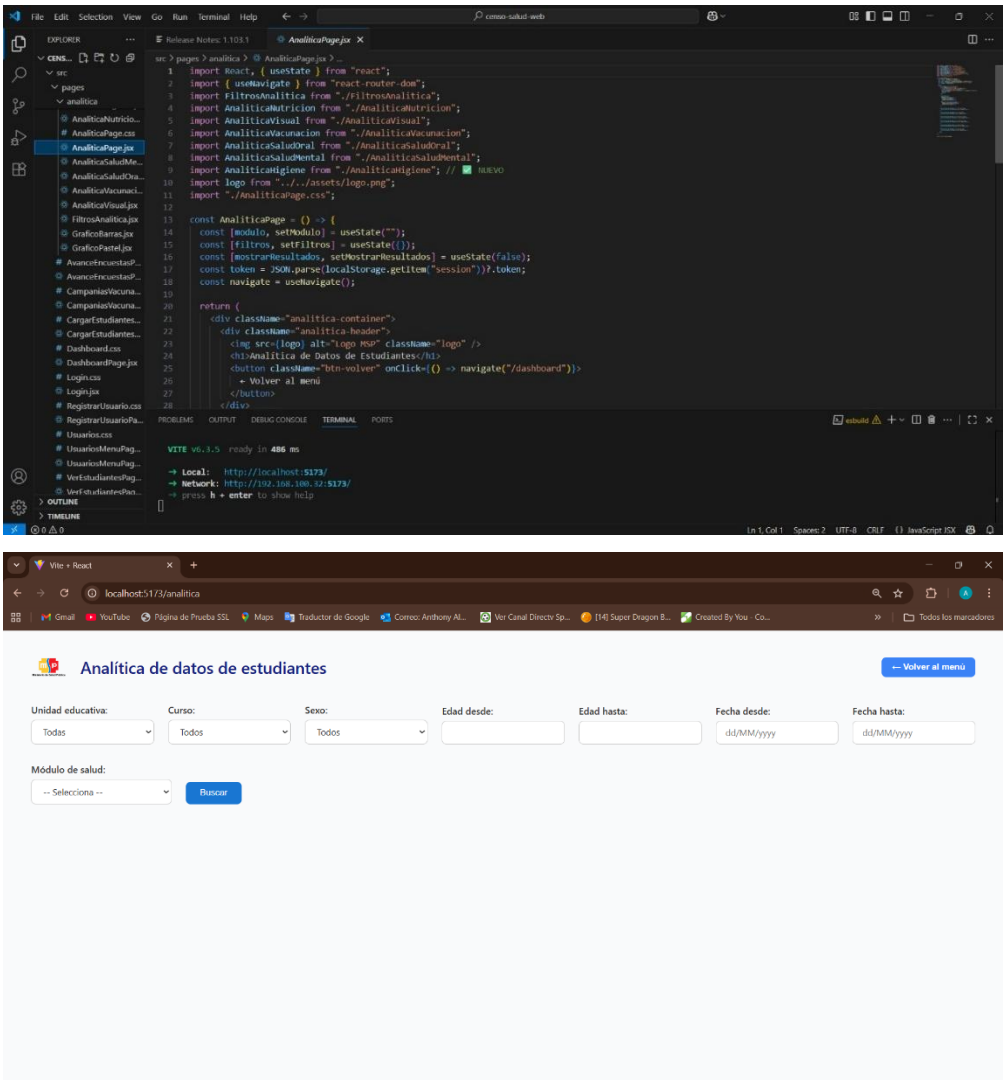


Figura 25: Módulo de gestión de analítica de datos (web)

El módulo que permite gestionar el estado de las encuestas se muestra en la Figura 26, desarrollado como mecanismo de seguimiento en tiempo real del progreso de las encuestas. Esta funcionalidad permite al administrador realizar búsquedas especificas aplicando filtros, mostrando los registros almacenados en la base de datos a través de peticiones al backend. Complementariamente, se agregó un indicador de progreso que detalla el estado de las encuestas (completadas y pendientes). Cabe destacar que también permite la exportación de los resultados en formato .xlxs.

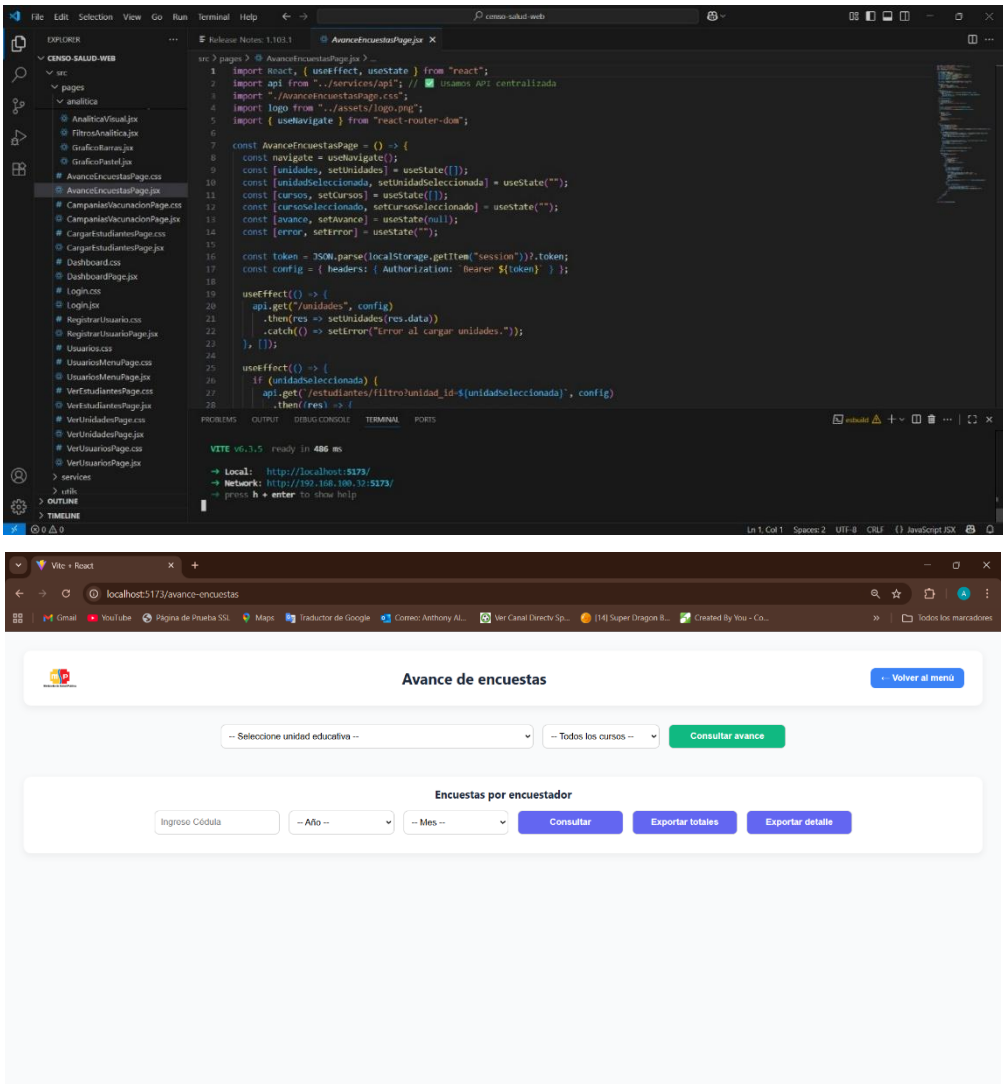


Figura 26: Módulo de reporte de encuestas (web)

La Figura 27 hace referencia a la carga de estudiantes, la misma que fue diseñada exclusivamente para el rol del administrador. Este módulo permite la incorporación de archivos en formato .xlsx, que son procesados mediante librerías como SheetJS y Axios. Los documentos Excel se validan para evitar duplicados o campos nulos antes de ser almacenados, quedando listos para su precarga en el aplicativo móvil.

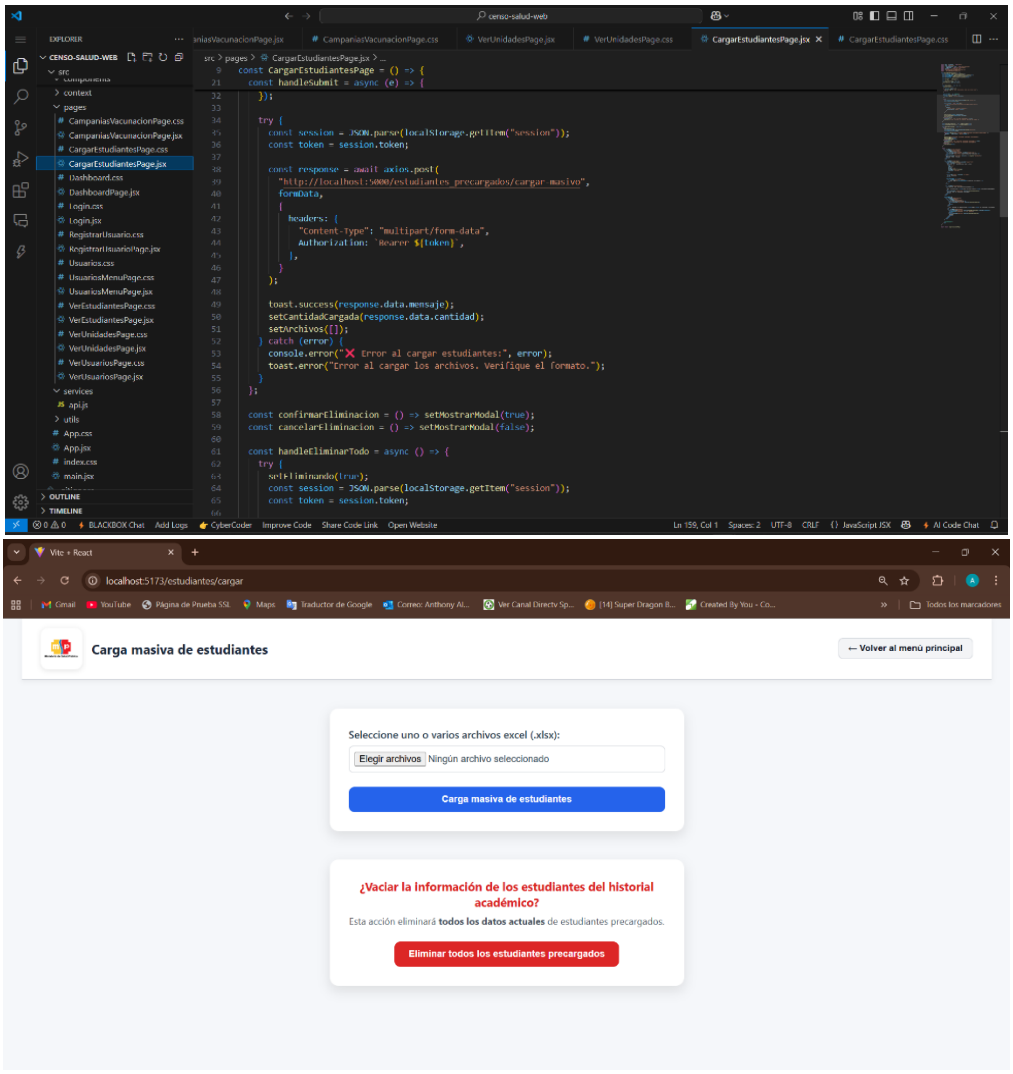


Figura 27: Carga de UV (web).

Fase 4: Pruebas del sistema

La fase final de la metodología Mobile-D hace referencia a las pruebas del sistema, siendo una etapa esencial para el desarrollo de aplicaciones, pues permitió verificar la eficiencia, precisión y disponibilidad de los datos.

- **Planificación de las pruebas**

En este apartado se planificaron distintos tipos de simulaciones con el objetivo de identificar y corregir errores o inconsistencias que puedan afectar a la fiabilidad de los datos, conforme con la norma ISO/IEC 25012:2008. Para la evaluación de los indicadores de eficiencia, precisión y disponibilidad de datos, se consideraron los valores referenciales que se detallan más adelante, correspondientes a cada uno de los indicadores mencionados.

a) Eficiencia de datos

El análisis de la eficiencia de los datos garantizo que la aplicación móvil gestione la información de manera ágil y segura. La planificación sobre este indicador se muestra en la Tabla 13.

Tabla 13: Planificación de la eficiencia de datos.

Categoría	Descripción
Pruebas realizadas	Ejecuciones de simulaciones de carga para medir eficiencia en la transferencia y sincronización de datos.
Herramientas utilizadas	Se hizo uso de Apache JMeter para realizar pruebas de rendimiento y carga en escenarios simulados.
Indicadores evaluados	• Tiempo de respuesta promedio: <=1 seg (por modulo). • Tasa promedio de transferencia de datos: >=50 KB/s

El valor de tiempo de respuesta promedio por módulo y la transferencia de datos, se obtuvo de Google SRE [35], en donde se menciona que, "ninguna solicitud puede recibir respuesta en menos de 0 ms, y un tiempo de espera de 1000 ms".

b) Precisión de los datos

La presión de los datos permitió evaluar el grado de exactitud con que el aplicativo almacena y procesa la información recolectada en los censos. La planificación sobre este indicador se muestra en la Tabla 14.

Tabla 14: Planificación de la precisión de los datos.

Categoría	Descripción
Pruebas realizadas	Se ejecutaron validaciones cruzadas entre los datos ingresados desde la aplicación móvil y los datos almacenados en el servidor.
Herramientas utilizadas	Se implemento la herramienta Python junto a la librería de Pandas para verificar la exactitud de los datos almacenados.
Indicadores evaluados	• Porcentaje de coincidencia de datos: >=95% • Porcentaje de errores: <=5%

El porcentaje considerado para la coincidencia de datos se basa en menciones dentro del artículo redactado en Tability [34], donde se menciona que, "una precisión de los datos del 95 % al 98 % garantiza la fiabilidad".

c) Disponibilidad

Este indicador permitió medir el acceso oportuno y continuo a los datos almacenados, asegurando así que la información se pueda visualizar y consultar sin interrupciones. La planificación sobre este indicador se muestra en la Tabla 15.

Tabla 15: Planificación de la disponibilidad.

Categoría	Descripción
Pruebas realizadas	Se realizaron pruebas de sincronización de datos, simulando una combinación de métodos POST y GET que consulta el servidor hasta obtener una respuesta valida.
Herramientas utilizadas	Se utilizo K6 para simular el tiempo que tarda la información en estar accesible para procesos de visualización.
Indicadores evaluados	• Tiempo promedio de propagación de la información: <=1seg

El valor de tiempo promedio de propagación de la información hace referencia al sitio web, donde las métricas se obtuvieron a partir de Core Web Vitals [33], mencionando que los sitios web LCP deben contar con un valor <= 2,5s para contar con un buen rendimiento.

- **Ejecución de pruebas**

Con el objetivo de garantizar la eficiencia, precisión y disponibilidad de los datos se aplicó la fase de ejecución de pruebas, donde se evaluaron distintos escenarios que permitieron medir el comportamiento del aplicativo de manera local bajo condiciones reales y simuladas.

a) Eficiencia de datos

Se realizaron pruebas de rendimiento enfocadas en medir el tiempo de respuesta y la transferencia de datos. Para ello, se utilizó la herramienta Apache JMeter, permitiendo simular escenarios de acceso concurrente. En total, se realizaron 25 simulaciones, cuyos resultados se muestran en la Figura 28, contando con un rango de 500 a 600 solicitudes simultaneas.

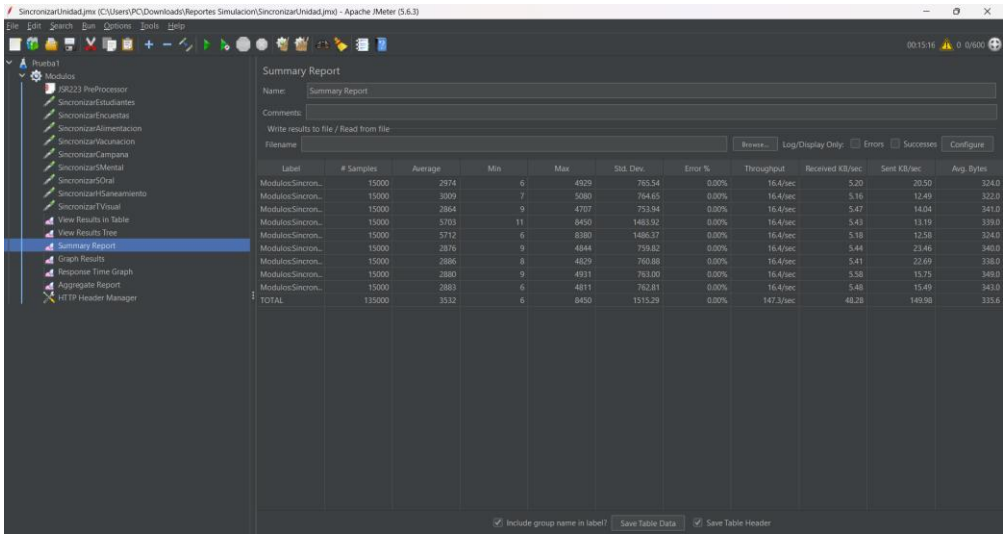


Figura 28: Tiempo de respuesta y tasa de transferencia de datos obtenido en JMeter.

b) Precisión de los datos

Las pruebas realizadas para este indicador permitieron verificar el porcentaje de coincidencia de datos y el porcentaje de errores. Para esto se desarrolló un script con el lenguaje de programación de alto nivel Python junto con la librería Pandas, especializada en la manipulación y análisis de datos. El desarrollo de este script facilitó realizar cálculos precisos sobre la exactitud de cada una de las tuplas presentes en las distintas tablas de la base de datos. En total se llevaron a cabo 25 simulaciones, cuyos resultados se presentan en la Figura 29.

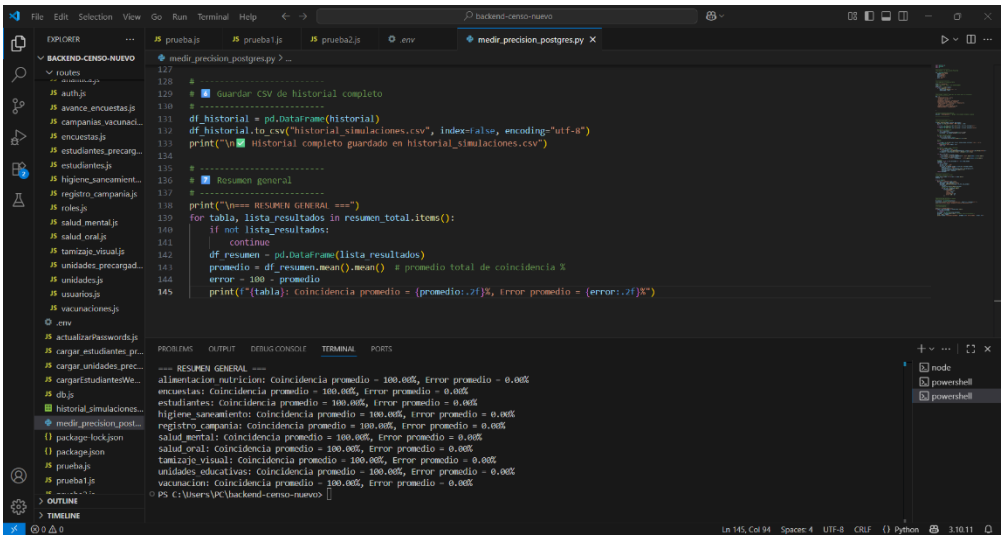


Figura 29: Porcentaje de coincidencia de datos y porcentaje de error obtenidos en Python.

c) Disponibilidad

Este indicador permitió evaluar el tiempo de propagación de la información, midiendo el lapso entra la carga de datos y su visualización en el apartado web. Como resultado de esta prueba realizada en la herramienta K6, la cual contó con 25 simulaciones, se obtuvieron los resultados que se presentan en la Figura 30, donde posterior a la carga de datos se recuperó la información de los estudiantes mediante su número de cédula.

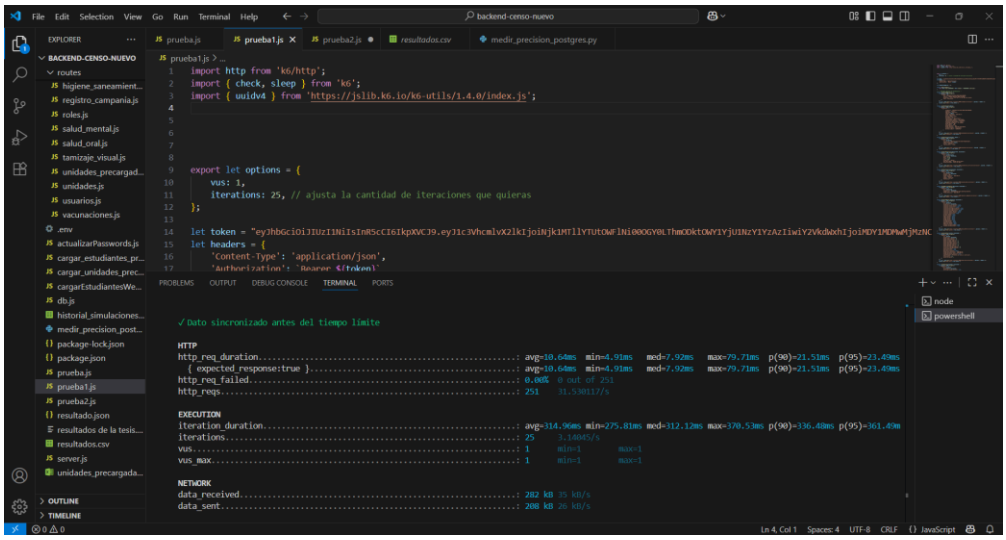


Figura 30: Tiempo de propagación de la información obtenido con K6.

Para ver el resultado de todas las simulaciones organizadas por indicador, véase el Anexo II.

CAPÍTULO IV. RESULTADOS Y DISCUSIÓN

4.1 Resultados

- **Eficiencia de datos**

El indicador de eficiencia de los datos permitió evaluar tanto el tiempo de respuesta como la capacidad de transferencia de información entre el aplicativo móvil y el servidor. Con el propósito de analizar la rapidez y estabilidad del sistema, se ejecutaron 25 pruebas de carga y transmisión de datos en JMeter, cuyos resultados se detallan en la Tabla 16.

Tabla 16: Resultados de la eficiencia de datos.

Código	Nombre de la prueba	Limite	Resultado
SED01	Tiempo de respuesta promedio (s)	<=1 s (por módulo)	0.35 s
SED03	Tasa promedio de transferencia de datos (KB/s)	>=50 KB/s	198,26 KB/s

A continuación, se ilustra el comportamiento de las dos métricas obtenidas a partir de las simulaciones realizadas en JMeter en la Figura 31 y Figura 32.

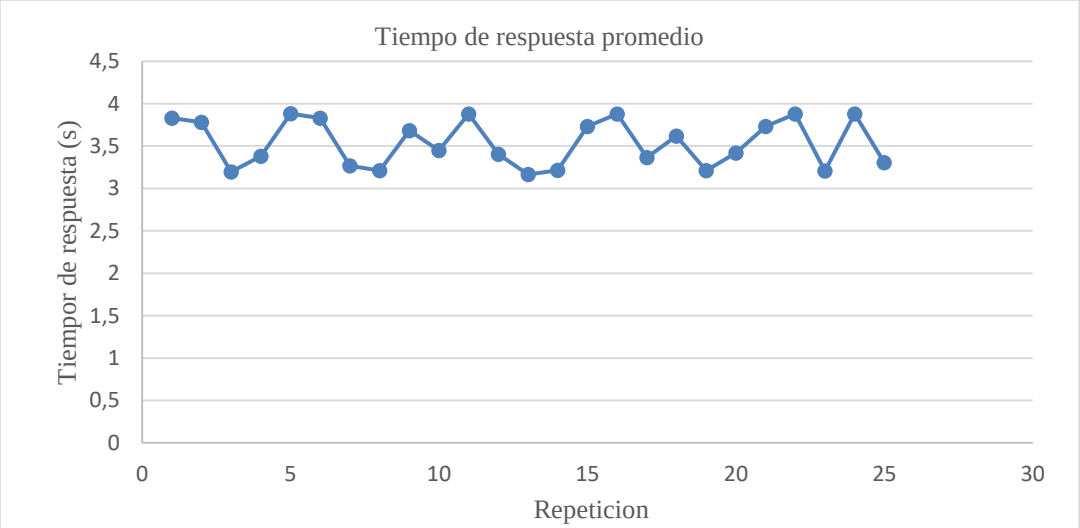


Figura 31: Comparación de resultados eficiencia de los datos (tiempo de respuesta).

Los resultados obtenidos en la dimensión de eficiencia muestran que el sistema presenta un tiempo de respuesta promedio de 3,53 segundos para la ejecución total de los 10 módulos, demostrando que cada módulo responde en aproximadamente 0,35 segundos. Este valor se encuentra holgadamente dentro del umbral establecido de ≤1 segundo por módulo, evidenciando un desempeño adecuado y consistente en términos de tiempo de respuesta.

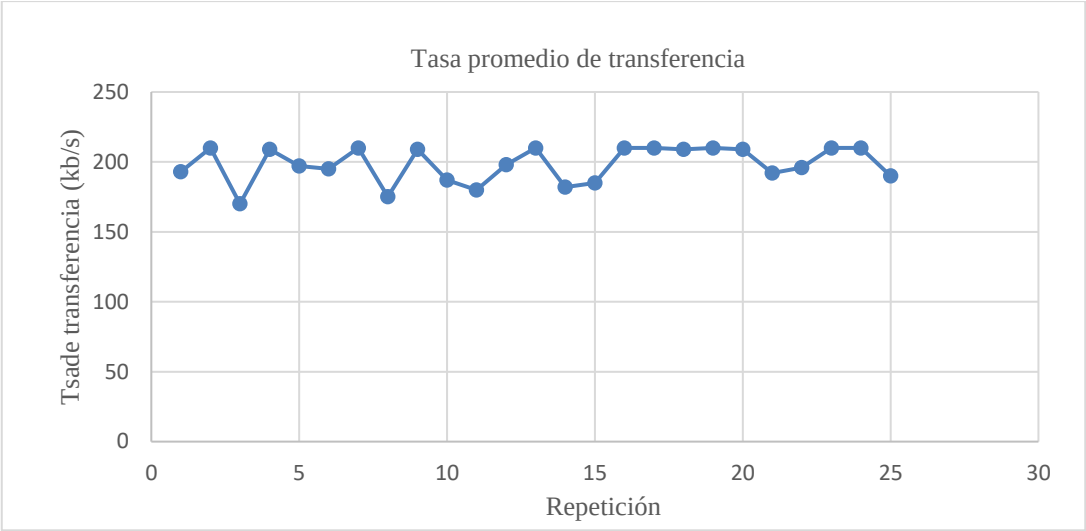


Figura 32: Comparación de resultados eficiencia de los datos (tasa de transferencia).

En cuanto a la tasa de transferencia de datos, se registraron valores comprendidos entre 170 KB/s y 210 KB/s, con un promedio cercano a 198,24 KB/s. Esta cifra supera ampliamente el criterio mínimo definido de ≥50 KB/s, garantizando que el sistema dispone de una capacidad suficiente de transmisión de datos para sostener operaciones de consulta y actualización de manera fluida.

En conjunto, estos resultados permiten concluir que la aplicación cumple satisfactoriamente con la dimensión de eficiencia definida en ISO/IEC 25012, asegurando que los datos pueden ser procesados y entregados de forma rápida y utilizando recursos adecuados. Desde la perspectiva de la fiabilidad, el cumplimiento de los objetivos de eficiencia contribuye a consolidar la confianza en el sistema, ya que los usuarios perciben una respuesta oportuna y estable en las operaciones, reduciendo la probabilidad de fallos relacionados con demoras o saturación en la transferencia de datos.

- **Precisión de los datos**

Este proceso permitió verificar que la información almacenada en el servidor coincidiera con los datos ingresados desde el aplicativo. Para evaluar este parámetro, se programó dentro del IDE Visual Studio Code, un script con el lenguaje de programación Python junto con su librería Pandas, permitiendo simular el envío de datos específicos y posteriormente validar que no existieran alteraciones ni pérdidas durante el proceso de sincronización. Los resultados obtenidos en las pruebas de precisión de los datos se presentan en la Tabla 17.

Tabla 17: Resultados de la precisión de los datos.

Código	Nombre de la prueba	Limite	Resultados
SPD01	Porcentaje de coincidencia de datos	>=95%	100%
SPD02	Porcentaje de error	<=5%	0%

La Figura 33 muestra una gráfica que resume la **proporción promedio obtenida por cada tabla** para los dos indicadores.

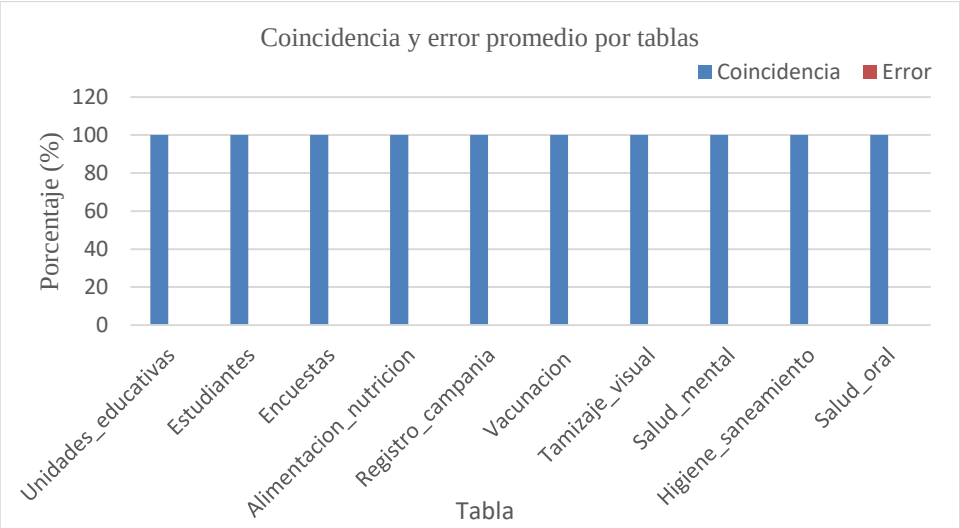


Figura 33: Porcentaje de precisión de los datos.

Los resultados obtenidos en la dimensión de precisión de los datos muestran que, para las 10 tablas evaluadas, se logró una coincidencia del 100% entre la información almacenada en el servidor y los datos ingresados desde el aplicativo móvil, sin registrarse ningún error de transmisión o alteración durante las 25 pruebas realizadas.

Este desempeño indica que el sistema garantiza la integridad de los datos durante el proceso de sincronización, asegurando que la información capturada por los usuarios se almacene de manera correcta y sin pérdidas. Desde la perspectiva de la ISO/IEC 25012, este cumplimiento refleja un alto nivel de precisión, contribuyendo de manera directa a la fiabilidad percibida del sistema, ya que los usuarios pueden confiar en que los datos que consultan o procesan son exactos y consistentes con la información original.

- **Disponibilidad**

El indicador de disponibilidad de los datos permitió evaluar la capacidad del sistema para mantener la información almacenada en el servidor accesible, así como garantizar la posibilidad de realizar consultas desde la plataforma web. Para ello, se ejecutó una simulación considerando el tiempo de propagación de la información, en la que se insertaron datos en el servidor mediante el método POST y posteriormente se verificó su accesibilidad a través de consultas utilizando el método GET. Los resultados obtenidos se presentan en la Tabla 18.

Tabla 18: Resultados de la disponibilidad.

Código	Nombre de la prueba	Limite	Resultados
SD01	Tiempo promedio de propagación de la información	<=1seg	112.84 ms

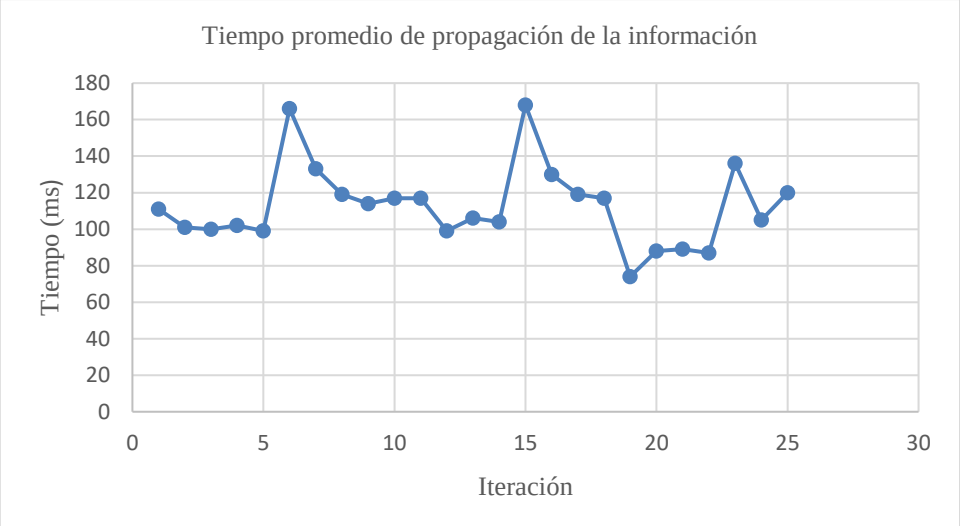


Figura 34: Tiempo de propagación de la información**.**

Durante la simulación realizada con K6, se registró un tiempo de propagación de la información de 112,84 ms. Este resultado evidencia que la sincronización de los datos entre el aplicativo y el servidor se realizó de manera efectiva, asegurando que la información estuviera accesible para su consulta en un tiempo mínimo. La medición refleja que el sistema mantiene un alto grado de disponibilidad, cumpliendo con la definición de la ISO/IEC 25012, al garantizar que los datos puedan ser obtenidos oportunamente por los usuarios y aplicaciones autorizadas.

Desde la perspectiva de la fiabilidad del sistema, este desempeño es idóneo ya que un tiempo de propagación reducido asegura que la información esté disponible de manera consistente y sin retrasos, minimizando la percepción de fallos y garantizando la continuidad operativa del aplicativo. Además, el valor obtenido evidencia una buena optimización de los procesos de sincronización y una infraestructura robusta que soporta la disponibilidad operativa del sistema, contribuyendo directamente a que los usuarios confíen en que los datos estarán accesibles cuando los necesiten.

Finalmente, la implementación del sistema web y móvil para la gestión de censos de salud en el Centro Salud Chambo tuvo un impacto positivo en la fiabilidad de los datos, al mejorar notablemente los criterios de eficiencia, precisión y disponibilidad definidos en la norma ISO/IEC 25012:2008. Los resultados demuestran que la digitalización del proceso reducirá errores de transcripción, optimización de tiempos de registro y permitir un acceso oportuno a la información, incluso en entornos sin conectividad gracias a la función offline. De esta forma, el sistema no solo fortaleció la calidad de los datos recolectados, sino que también garantizó su correcta utilización para la toma de decisiones en la gestión de la salud.

4.2 Discusión

En el presente proyecto de investigación, se evaluó la fiabilidad de los datos en base a la norma ISO/IEC 25012:2008, donde se establecen lineamientos específicos para evaluar las propiedades de un producto de datos determinado, priorizando la eficiencia, precisión y disponibilidad de la información.

Al comparar los resultados alcanzados con investigaciones similares, tales como “Aplicación web progresiva para el manejo de inventario en la farmacia de la coordinación de salud zona 3” [36], se observó que el aplicativo registro tiempos de respuesta entre 300 ms y 345 ms bajo escenarios de carga menores. En cambio, este proyecto manejo de 500 a 600 usuarios virtuales y procesos de sincronización más complejos, obteniendo un tiempo promedio de 350 ms en JMeter, si bien este valor supera ligeramente al proporcionado por el estudio comparado, las diferencias se justifican por la mayor cantidad de peticiones y la complejidad de las operaciones.

A lo mencionado anteriormente, se suma una ventaja significativa respecto a otros proyectos similares, como el proyecto titulado “Desarrollo de una aplicación web para el censo obstétrico y control neonatal en el Centro de Salud Chambo” [37], donde se desarrolló una aplicación web sin integrar funcionalidades offline. A diferencia del proyecto citado anteriormente, el sistema planteado en esta investigación permite al usuario almacenar los datos recolectados de manera local temporalmente, es decir, hasta que el usuario disponga de conexión a internet y pueda sincronizar la información recolectada con el servidor.

Finalmente, es importante mencionar que, a diferencia de las investigaciones analizadas previamente, tanto Tapuy y Segovia [36] como Montaño [37] se centraron únicamente en el desarrollo de un sitio web, el mismo que servía para recolectar y analizar la información. La solución propuesta en esta investigación está conformada por dos aplicativos complementarios: una aplicación móvil, encargada de recolectar y enviar la información al servidor, y una aplicación web, destinada a la visualización, gestión y análisis de los datos recolectados.

CAPÍTULO V. CONCLUSIONES y RECOMENDACIONES

5.1 Conclusiones

- Tras la investigación y aplicación de la metodología Mobile-D, se logró comprobar su efectividad en el desarrollo de sistemas multiplataforma por su enfoque ágil, además, gracias a su estructura iterativa se implementaron las distintas funcionalidades de forma ordenada y eficiente. La metodología Mobile-D fue la que mejor se ajustó al desarrollo del proyecto, ya que facilitó una adaptación progresiva a los requerimientos funcionales, asegurando una correcta sincronización de la aplicación móvil y web destinada a la recolección y análisis de datos en el centro de salud Chambo.
- La creación de un sistema para la realización de censos de salud mediante dispositivos móviles, junto con el desarrollo de una plataforma web, se llevó a cabo satisfactoriamente siguiendo la metodología ágil Mobile-D. La aplicación móvil desarrollada mediante el framework Flutter, facilita la recolección de información de manera offline para su posterior sincronización, en base a módulos que se enfocan en distintas áreas de la salud. Por su parte, la plataforma web realizada con Node,js y React, posibilitó la gestión, visualización de reportes y análisis de datos recolectados, brindando así a los administrativos del centro de salud Chambo, una herramienta para la toma de decisiones basadas en información oportuna y precisa.
- Los resultados obtenidos en base a los criterios de la norma ISO/IEC 25012:2008 definidos en base a características de calidad que deben considerarse en la evaluación de un producto enfocado en los datos permitieron verificar que el sistema mantiene un alto nivel de fiabilidad, evaluada principalmente a través de tres indicadores clave: eficiencia, precisión y disponibilidad. Las simulaciones se realizaron en JMeter y mediante scripts programados en K6 y Python, obteniendo tiempos de respuesta promedio de 3,53 seg, además, se evidencio un porcentaje de error del 0% en el aparatado de la consistencia de los datos. Finalmente, un valor muy por debajo del límite en base a la propagación de la información, siendo este de 112,84 ms. Estos resultados permiten concluir en una solución confiable y ágil para la gestión de censos de salud, cumpliendo con los estándares de calidad de datos definidos por la norma ISO y aportando beneficios gracias a su enfoque multiplataforma.

5.2 Recomendaciones

- Se sugiere continuar utilizando metodologías agiles como Mobile-D para futuros proyectos multiplataforma, dado a su capacidad de adaptación a requerimientos cambiantes y su enfoque iterativo durante el desarrollo.
- Se recomienda la actualización y expansión de funcionalidades del aplicativo móvil y web, que incluyan nuevos módulos de salud y reportes automatizados, potenciando aún más la toma de decisiones.
- Se aconseja realizar evaluaciones periódicas de los indicadores de calidad de los datos definidos por la norma ISO/IEC 25012:2008 para mantener la fiabilidad. Además, en caso de expandir el sistema, se deben ajustar los valores limite conforme crezca el volumen y diversidad de la información gestionada.

BIBLIOGRAFÍA

[1] Organización Mundial de la Salud, «Global strategy on digital health 2020-2025,» 2021. [En línea]. Available: https://www.who.int/docs/default-source/documents/gs4dhdaa2a9f352b0445bafbc79ca799dce4d.pdf.

[2] Ministerio de Salud Pública del Ecuador, «Informe de Gestión 2022,» 2022. [En línea]. Available: https://www.salud.gob.ec/wp-content/uploads/2023/06/5.2FASE-2_INFORME_FINAL_-RC_2022_CZ5-1.pdf.

[3] Ministerio de salud pública, «Registro oficial - Tercer Suplemento N°715,» 06 01 2025. [En línea]. Available: https://www.hgdc.gob.ec/images/Hospital/Base%20legal/AC-00068-2024%20DIC%2018-Tercer%20Suplemento%20Nro.%20715%20Politica%20de%20Transformacion%20Digital%20de%20la%20Salud.pdf. [Último acceso: 11 08 2025].

[4] Public Health International, «Open Access Pub,» [En línea]. Available: https://openaccesspub.org/public-health-international/census?utm_source=chatgpt.com.

[5] M. Colwill y A. Poullis, «Using national census data to facilitate healthcare research,» 13 12 2023. [En línea]. Available: https://doi.org/10.5662/wjm.v13.i5.414. [Último acceso: 15 08 2025].

[6] J. Baldoceta, «Desarrollo de un aplicativo móvil basado en la metodología Mobile-D para la gestión de reservas del hotel Caribe de Huaral.,» 2017. [En línea]. Available: https://repositorio.uigv.edu.pe/backend/api/core/bitstreams/e6507d8c-6a2f-45db-bdf5-2245d0a3ccf8/content. [Último acceso: 14 03 2025].

[7] H. Huaraca, «Uso de aplicaciones móviles en el proceso de aprendizaje del idioma inglés por parte de los estudiantes de la Carrera de Pedagogía de los Idiomas Nacionales y Extranjeros de la UNACH.,» 2022. [En línea]. Available: http://dspace.unach.edu.ec/bitstream/51000/9697/1/UNACH-EC-FCEHT-PCEINF-0009-2022.pdf. [Último acceso: 11 03 2025].

[8] E. Guaman y E. Yambay, «Desarrollo de una aplicación web y móvil utilizando la metodología agile inception, para la gestión de servicios de trabajos informales en Riobamba,» 2022. [En línea]. Available:

http://dspace.unach.edu.ec/bitstream/51000/10091/1/Guaman%20S.%2C%20Edwin%20A.%20Yambay%20L.%2C%20Edison%20F.%20(2022)%20Desarrollo%20de%20una%20aplicación%20web%20y%20móvil%20utilizando%20la%20metodología%20agile%20inception%2C%20para%20la%20gestión%20de. [Último acceso: 12 03 2025].

[9] L. Vallejo, «Desarrollo de una aplicación móvil para la adopción de animales abandonados centro de rescate animal (CRIAR) Municipio de Riobamba.,» 02 05 2024. [En línea]. Available: http://dspace.espoch.edu.ec/handle/123456789/22008.

[10] E. Valdivieso, «DISEÑO E IMPLEMENTACIÓN DE UNA APLICACIÓN TURÍSTICA DE LA CIUDAD DE RIOBAMBA PARA DISPOSITIVOS CON SISTEMA OPERATIVO ANDROID,» 2016. [En línea]. Available: http://dspace.unach.edu.ec/bitstream/51000/2951/1/UNACH-FCEHT-TG-INFORM-2016-000019.pdf. [Último acceso: 12 03 2025].

[11] Blog Tecnologia Movil , «Arquitectura de Android,» [En línea]. Available: https://tecnologiamovil128806266.wordpress.com/equipos/. [Último acceso: 12 03 2025].

[12] V. Vázquez, «Desarrollo de aplicaciones móviles multiplataforma con Flutter,» 2019. [En línea]. Available: https://repositorio.ual.es/bitstream/handle/10835/8010/TFG_VAZQUEZ%20RODRIGUEZ%2c%20VICTOR.pdf?sequence=1&isAllowed=y. [Último acceso: 11 03 2025].

[13] J. Baldrés, Desarrollo de una aplicación multiplataforma mediante el framework Flutter e implementación de servicios de autenticación y base de datos mediante Firebase, Valencia: Universitat Politecnica de Valencia, 2020.

[14] Google, «Descripción general de la arquitectura de Flutter,» Flutter Documentation , [En línea]. Available: https://docs.flutter.dev/resources/architectural-overview. [Último acceso: 11 03 2025].

[15] K. Escobar, «Desarrollo de la aplicación móvil de gestión de cupones en las campañas de mercadotecnia del diario Elespecial-noticias para influir en su eficiencia,» 20 11 2024. [En línea]. Available: http://dspace.espoch.edu.ec/handle/123456789/23168.

[16] S. Avalos y D. Guaillas, «APLICACIÓN WEB Y MÓVIL PARA LA GESTIÓN DE SERVICIOS GASTRONÓMICOS DE RESTAURANTES DEL CANTÓN

SARAGURO.,» 2023. [En línea]. Available: http://dspace.unach.edu.ec/handle/51000/11359. [Último acceso: 18 05 2025].

[17] C. Zuñiga, «Software de recorrido virtual para pymes con una vitrina web del cantón Guano utilizando la tecnología Three.js,» 2024. [En línea]. Available: http://dspace.unach.edu.ec/handle/51000/13947. [Último acceso: 18 05 2025].

[18] T. Brito, M. Ferreira, M. Monteiro, P. Lopes, M. Barros, J. Santos y N. Santos, «Study of JavaScript Static Analysis Tools for,» 2023. [En línea]. Available: https://arxiv.org/pdf/2301.05097. [Último acceso: 11 03 2025].

[19] SQLite, «¿Qué es SQLite?,» [En línea]. Available: https://sqlite.org/. [Último acceso: 12 03 2025].

[20] C. Salcan, «Prototipo de un Sistema en GNS3 con la Integración de Asterisk y PostgreSQL sobre IPv6 para Consulta de Notas Académicas,» 2024. [En línea]. Available: http://dspace.unach.edu.ec/handle/51000/13176. [Último acceso: 18 05 2025].

[21] L. Zapata, «ANALÍTICA DE DATOS PARA MONITOREAR EL DESEMPEÑO DE LOS ESTUDIANTES DEL COLEGIO FRANCISCO DE MIRANDA.,» 2021. [En línea]. Available: https://hdl.handle.net/10983/27216. [Último acceso: 12 03 2025].

[22] P. Abrahamsson, A. Hanhineva, H. Hulkko, T. Ihme, J. Jäälinoja, M. Korkala, J. Koskela, P. Kyllönen y O. Salo, «Mobile-D: An Agile Approach for Mobile Application Development,» 20 09 2017. [En línea]. Available: https://doi.org/10.48550/arXiv.1709.06820. [Último acceso: 12 08 2025].

[23] A. Kaur y K. Kaur, «Systematic literature review of mobile application development and testing effort estimation,» 02 2022. [En línea]. Available: https://doi.org/10.1016/j.jksuci.2018.11.002. [Último acceso: 12 08 2025].

[24] P. Ramírez, «Implementación de un aplicativo móvil y web para la gestión administrativa de la empresa constsa usando la metodología mobile-d,» 2021. [En línea]. Available: http://repositorio.utmachala.edu.ec/handle/48000/17863. [Último acceso: 13 11 2025].

[25] C. Muñoz, «APLICACIÓN DE LA METODOLOGÍA MOBILE-D EN EL DESARROLLO DE UNA APP MÓVIL PARA GESTIONAR CITAS MÉDICAS DEL CENTRO JEL RIOBAMBA,» 2020. [En línea]. Available:

http://dspace.unach.edu.ec/bitstream/51000/7073/2/7.%20APLICACIÓN%20DE%20LA%20METODOLOGÍA%20MOBILE-D%20EN%20EL%20DESARROLLO%20DE%20UNA%20APP%20MÓVIL%20PARA%20GESTIONAR%20CITAS%20MÉDICAS%20DEL%20CENTRO%20JEL%20RIOBAMBA.pdf. [Último acceso: 14 03 2025].

[26] S. Márquez, «SISTEMA COMPUTACIONAL PARA ESTIMAR LA CAPTURA DE CARBONO EN AGROECOSISTEMAS DE CAFÉ: CASO HUATUSCO, VERACRUZ,» 2016. [En línea]. Available: https://www.researchgate.net/figure/Figura-210-Metodologia-Mobile-D-Fuente-Leyva-et-al-2016_fig5_348295603. [Último acceso: 08 10 2025].

[27] F. Sepa, «DESARROLLO DE UNA APLICACIÓN MÓVIL ANDROID PARA LA PROMOCIÓN Y DIFUSIÓN DE EVENTOS Y TURISMO DEL CANTÓN GUARANDA UTILIZANDO LA METODOLOGÍA DE DESARROLLO MOBILE-D,» 2022. [En línea]. Available: http://dspace.espoch.edu.ec/handle/123456789/20775. [Último acceso: 14 03 2025].

[28] A. Minina, «DESARROLLO DE UNA APLICACIÓN MÓVIL DELIVERY PARA VISUALIZACIÓN DEL MENÚ Y REALIZACIÓN PEDIDOS DE COMIDAS Y BEBIDAS A DOMICILIO EN EL BARRESTAURANTE BALTIMORE,» 2023. [En línea]. Available: http://dspace.espoch.edu.ec/handle/123456789/22427. [Último acceso: 14 03 2025].

[29] ISO 25000, «ISO/IEC 25012,» [En línea]. Available: https://iso25000.com/index.php/normas-iso-25000/iso-25012. [Último acceso: 11 03 2025].

[30] F. Gualo, M. Rodríguez, J. Verdugo, I. Caballero y M. Piattini, «Data Quality Certification using ISO/IEC 25012: Industrial Experiences,» 23 02 2021. [En línea]. Available: https://doi.org/10.48550/arXiv.2102.11527.

[31] S. Cobos y J. Maigua, «Desarrollo de una aplicación web para el análisis de los datos de estabilidad del carbono orgánico en la zona alto andina de la sierra centro del Ecuador.,» 10 05 2024. [En línea]. Available: http://dspace.espoch.edu.ec/handle/123456789/22071.

[32] J. Wang, Y. Liu, P. Li, Z. Lin, S. Sindakis y S. Aggarwal, «Overview of Data Quality: Examining the Dimensions, Antecedents, and Impacts of Data Quality,» 10 02 2023.

[En línea]. Available: https://doi.org/10.1007/s13132-022-01096-6. [Último acceso: 14 11 2025].

[33] Google, «Core Web Vitals,» 17 02 2025. [En línea]. Available: https://developers.google.com/search/docs/appearance/core-web-vitals?hl=es. [Último acceso: 23 06 2025].

[34] M. Meucci y A. Muller, OWASP Web Security Testing Guide v4, Wilmington: The OWASP Foundation, 2020.

[35] Google, «Google SRE,» 2017. [En línea]. Available: https://sre.google/sre-book/service-level-objectives/. [Último acceso: 23 06 2025].

[36] F. Tapuy y S. Segovia, «APLICACIÓN WEB PROGRESIVA PARA EL MANEJO DE INVENTARIO EN LA FARMACIA DE LA COORDINACIÓN DE SALUD ZONA 3,» 27 05 2022. [En línea]. Available: http://dspace.unach.edu.ec/handle/51000/9159. [Último acceso: 01 07 2025].

[37] R. Montaño, «Desarrollo de una aplicación web para el censo obstétrico y control neonatal en el Centro de Salud Chambo,» 16 05 2024. [En línea]. Available: http://dspace.unach.edu.ec/handle/51000/12950. [Último acceso: 02 07 2025].

[38] Google, « Flutter documentation,» Flutter, [En línea]. Available: https://docs.flutter.dev/. [Último acceso: 10 03 2025].

[39] T. Gironés, «Arquitectura de Android,» Valencia, 2011.

[40] C. Peralta, «INTELIGENCIA DE NEGOCIOS APLICADA A LA GESTIÓN ESTRATÉGICA DE INFORMACIÓN COMERCIAL, DENTRO DEL PROCESO DE TOMA DE DECISIONES EN VENTAS DE PYMES,» 2022. [En línea]. Available: http://dspace.unach.edu.ec/bitstream/51000/9033/1/Proyecto%20de%20Investigación.pdf. [Último acceso: 12 03 2025].

[41] H. Ghalavand, S. Shirshahi, A. Rahimi, Z. Zarrinabadi y F. Amani, «Common data quality elements for health information systems: a systematic review,» 02 09 2024. [En línea]. Available: https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-024-02644.

ANEXOS

Anexo I: Diccionario de la base de datos

a) Tabla de usuarios

En la Tabla 19 se hace referencia a todos los campos que contiene la tabla usuarios.

Tabla 19: Diccionario de la tabla usuarios.

Campo	Tipo de Dato	Descripción
id	uuid	Identificador único del usuario
cedula	varchar(10)	Cédula del usuario
nombres	varchar(100)	Nombres del usuario
apellidos	varchar(100)	Apellidos del usuario
password	text	Contraseña encriptada
rol_id	uuid	Rol del usuario (FK)
fecha_creacion	timestamp	Fecha de creación
carrera	varchar(100)	Carrera del usuario
fecha_nacimiento	date	Fecha de nacimiento
genero	varchar(20)	Género
area_trabajo	varchar(100)	Área de trabajo
estado	varchar	Estado del usuario (Activo/Inactivo)

b) Tabla de roles

En la Tabla 20 se hace referencia a todos los campos que contiene la tabla roles de usuario.

Tabla 20: Diccionario de la tabla roles.

Campo	Tipo de Dato	Descripción
id	uuid	Identificador del rol.
nombre	varchar(50)	Nombre del rol (admin, encuestador, odontólogo, rector).

c) Tabla de unidades educativas

En la Tabla 21 se hace referencia a todos los campos que contiene la tabla unidades educativas.

Tabla 21: Diccionario de la tabla unidades educativas.

Campo	Tipo de Dato	Descripción
id	uuid	Identificador de la unidad educativa.
nombre	varchar(150)	Nombre de la unidad.
direccion	text	Dirección.
tipo	varchar(50)	Tipo de unidad educativa.
fecha_creacion	timestamp	Fecha de creación.

d) Tabla de unidades precargadas

En la Tabla 22 se hace referencia a todos los campos que contiene la tabla unidades precargadas.

Tabla 22: Diccionario de la tabla unidades precargadas.

Campo	Tipo de Dato	Descripción
id	uuid	Identificador único.
nombre	varchar(255)	Nombre de la unidad educativa.
direccion	varchar(255)	Dirección.
tipo	varchar(100)	Tipo de unidad educativa.

e) Tabla de estudiantes

En la Tabla 23 se hace referencia a todos los campos que contiene la tabla estudiante.

Tabla 23: Diccionario de la tabla estudiantes.

Campo	Tipo de Dato	Descripción
cedula	varchar(10)	Cédula del estudiante.
unidad_id	uuid	Unidad educativa (FK).
nombres	varchar(100)	Nombres.
apellidos	varchar(100)	Apellidos.
fecha_nacimiento	date	Fecha de nacimiento.
sexo	varchar(10)	Sexo.
discapacidad	varchar(50)	Tipo de discapacidad.
alergias	text	Alergias.
representante_nombre	varchar(100)	Nombre del representante.
representante_telefono	varchar(20)	Teléfono del representante.
comunidad_residencia	varchar(100)	Comunidad de residencia.
fecha_registro	timestamp	Fecha de registro.
profesor_nombre	varchar(100)	Nombre del profesor.
profesor_cedula	varchar(20)	Cédula del profesor.
tipo_alergia	text	Tipo de alergia.
parentesco_representante	text	Parentesco del representante.
posee_seguro	text	Posee seguro.
tipo_seguro	text	Tipo de seguro.
grado_paralelo	text	Grado y paralelo.

f) Tabla de estudiantes precargados

En la Tabla 24 se hace referencia a todos los campos que contiene la tabla estudiante precargados.

Tabla 24: Diccionario de la tabla estudiantes precargados.

Campo	Tipo de Dato	Descripción
cedula	varchar(20)	Cédula del estudiante.
nombres	varchar(100)	Nombres.
apellidos	varchar(100)	Apellidos.
fecha_nacimiento	date	Fecha de nacimiento.
sexo	varchar(10)	Sexo.
discapacidad	varchar(50)	Tipo de discapacidad.
alergias	text	Alergias.
nombre_representante	varchar(100)	Nombre del representante.
telefono_representante	varchar(20)	Teléfono del representante.
comunidad_residencia	varchar(100)	Comunidad de residencia.
profesor_nombre	varchar(100)	Nombre del profesor.
profesor_cedula	varchar(20)	Cédula del profesor.
tipo_alergia	text	Tipo de alergia.
parentesco_representante	text	Parentesco del representante.
posee_seguro	text	Posee seguro.
tipo_seguro	text	Tipo de seguro.
grado_paralelo	text	Grado y paralelo.

g) Tabla de encuestas

En la Tabla 25 se hace referencia a todos los campos que contiene la tabla encuestas.

Tabla 25: Diccionario de la tabla encuestas.

Campo	Tipo de Dato	Descripción
id	uuid	Identificador único.
usuario_id	uuid	ID del usuario que registró (FK).
cedula_estudiante	varchar(10)	Cédula del estudiante evaluado.
fecha	timestamp	Fecha de creación.
tipo_encuesta	varchar(50)	Tipo de encuesta (salud, salud_oral).

h) Tabla de alimentación y nutrición

En la Tabla 26 se hace referencia a todos los campos que contiene la tabla alimentación y nutrición.

Tabla 26: Diccionario de la tabla alimentación y nutrición.

Campo	Tipo de Dato	Descripción
id	uuid	Identificador del registro.
encuesta_id	uuid	ID de encuesta (FK).
peso	numeric(5,2)	Peso en kg.
talla	numeric(5,2)	Talla en metros.

imc	numeric(5,2)	Índice de masa corporal.
estado_imc	varchar(50)	Estado nutricional por IMC.
peso_para_edad	varchar(50)	Clasificación de peso para edad.
talla_para_edad	varchar(50)	Clasificación de talla para edad.

i) Tabla de vacunación

En la Tabla 27 se hace referencia a todos los campos que contiene la tabla vacunación.

Tabla 27: Diccionario de la tabla vacunación.

Campo	Tipo de Dato	Descripción
id	uuid	Identificador.
encuesta_id	uuid	ID de encuesta (FK).
vacuna	varchar(50)	Nombre de la vacuna.
fecha_vacunacion	date	Fecha de vacunación.
tipo	varchar(50)	Tipo de vacuna (regular o campaña).
estado	varchar(50)	Estado de aplicación.
observacion	text	Observaciones.

j) Tabla de registro de campañas

En la Tabla 28 se hace referencia a todos los campos que contiene la tabla registro de campañas.

Tabla 28: Diccionario de la tabla registro de campañas.

Campo	Tipo de Dato	Descripción
id	uuid	ID del registro.
encuesta_id	uuid	ID de encuesta (FK).
campania_id	uuid	ID de campaña (FK).
fecha_vacunacion	text	Fecha de vacunación.
created_at	timestamp	Fecha de creación.

k) Tabla de tamizaje visual

En la Tabla 29 se hace referencia a todos los campos que contiene la tabla tamizaje visual.

Tabla 29: Diccionario de la tabla tamizaje visual.

Campo	Tipo de Dato	Descripción
id	uuid	Identificador del registro.
encuesta_id	uuid	ID de encuesta (FK).
ojo_izquierdo_sin_lentes	varchar(30)	Agudeza visual ojo izquierdo sin lentes.
ojo_derecho_sin_lentes	varchar(30)	Agudeza visual ojo derecho sin lentes.
ojo_izquierdo_con_lentes	varchar(30)	Agudeza visual ojo izquierdo con lentes.
ojo_derecho_con_lentes	varchar(30)	Agudeza visual ojo derecho con lentes.

resultado_sin_lentes	text	Resultado general sin lentes.
resultado_con_lentes	text	Resultado general con lentes.

l) Tabla de salud oral

En la Tabla 30 se hace referencia a todos los campos que contiene la tabla salud oral.

Tabla 30: Diccionario de la tabla salud oral.

Campo	Tipo de Dato	Descripción
id	uuid	Identificador del registro.
encuesta_id	uuid	ID de encuesta (FK).
placa_bacteriana	boolean	Presencia de placa bacteriana.
aplicacion_fluor	boolean	Aplicación de flúor.
aplicacion_sellantes	boolean	Aplicación de sellantes.
fecha_placa_bacteriana	date	Fecha de control de placa bacteriana.
fecha_aplicacion_fluor	date	Fecha de aplicación de flúor.
fecha_aplicacion_sellantes	date	Fecha de aplicación de sellantes.
observaciones	text	Observaciones del odontólogo.
piezas_sanas_superior	integer	Piezas sanas del maxilar superior.
piezas_sanas_inferior	integer	Piezas sanas del maxilar inferior.
piezas_cariadas_superior	integer	Piezas cariadas del maxilar superior.
piezas_cariadas_inferior	integer	Piezas cariadas del maxilar inferior.
piezas_obturadas_superior	integer	Piezas obturadas del maxilar superior.
piezas_obturadas_inferior	integer	Piezas obturadas del maxilar inferior.
piezas_perdidas_superior	integer	Piezas perdidas del maxilar superior.
piezas_perdidas_inferior	integer	Piezas perdidas del maxilar inferior.
extraccion_indicada_superior	text	Piezas con extracción indicada superior.
extraccion_indicada_inferior	text	Piezas con extracción indicada inferior.

m) Tabla de salud mental

En la Tabla 31 se hace referencia a todos los campos que contiene la tabla salud mental.

Tabla 31: Diccionario de la tabla salud mental.

Campo	Tipo de Dato	Descripción
id	uuid	Identificador del registro.
encuesta_id	uuid	ID de encuesta (FK).
violencia_fisica_familiar	boolean	Violencia física en el entorno familiar.
violencia_fisica_escolar	boolean	Violencia física en el entorno escolar.
violencia_fisica_otro	boolean	Violencia física en otros contextos.
violencia_psicologica_familiar	boolean	Violencia psicológica familiar.
violencia_psicologica_escolar	boolean	Violencia psicológica escolar.
violencia_psicologica_otro	boolean	Violencia psicológica en otro ámbito.

violencia_sexual_familiar	boolean	Violencia sexual en el hogar.
violencia_sexual_escolar	boolean	Violencia sexual en la escuela.
violencia_sexual_otro	boolean	Violencia sexual en otros espacios.
alcohol_pasado	boolean	Consumo de alcohol en el pasado.
alcohol_actual	boolean	Consumo actual de alcohol.
tabaco_pasado	boolean	Consumo de tabaco en el pasado.
tabaco_actual	boolean	Consumo actual de tabaco.
drogas_pasado	boolean	Consumo de drogas en el pasado.
drogas_actual	boolean	Consumo actual de drogas.
ansiedad_pasada	boolean	Síntomas de ansiedad en el pasado.
ansiedad_actual	boolean	Síntomas actuales de ansiedad.
depresion_pasada	boolean	Síntomas de depresión en el pasado.
depresion_actual	boolean	Síntomas actuales de depresión.
intento_suicidio_pasado	boolean	Intentos de suicidio en el pasado.
intento_suicidio_actual	boolean	Intentos actuales de suicidio.

n) Tabla de higiene y saneamiento

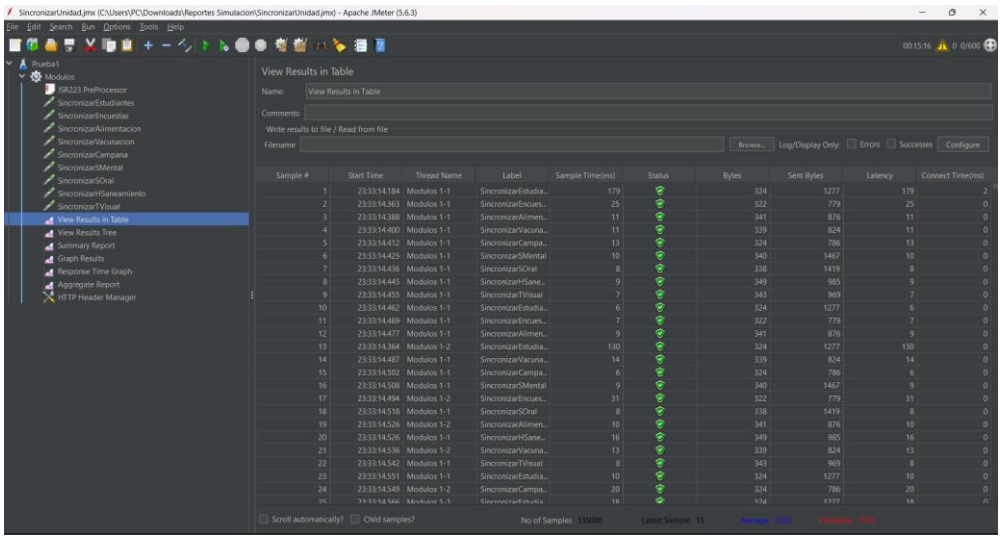
En la Tabla 32 se hace referencia a todos los campos que contiene la tabla higiene y saneamiento.

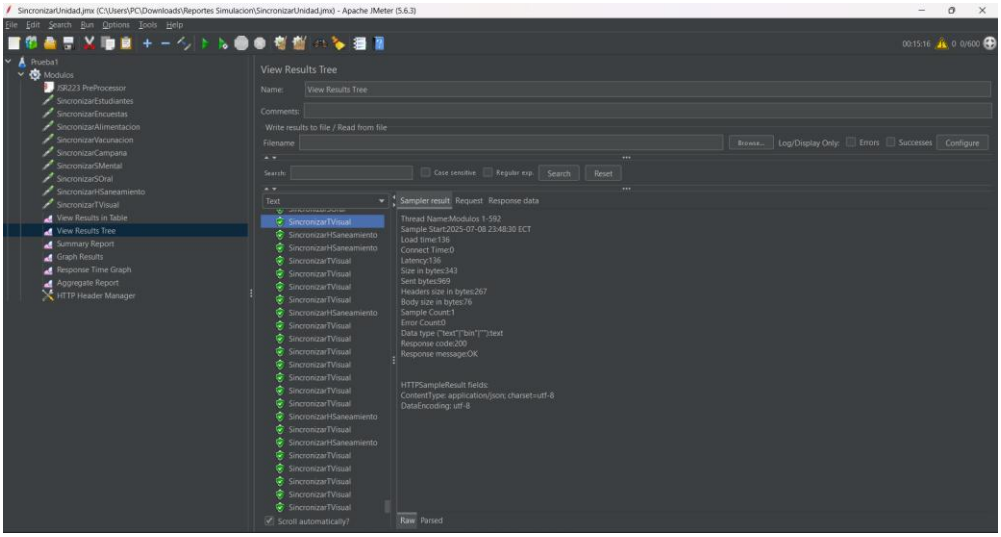
Tabla 32: Diccionario de la tabla higiene y saneamiento.

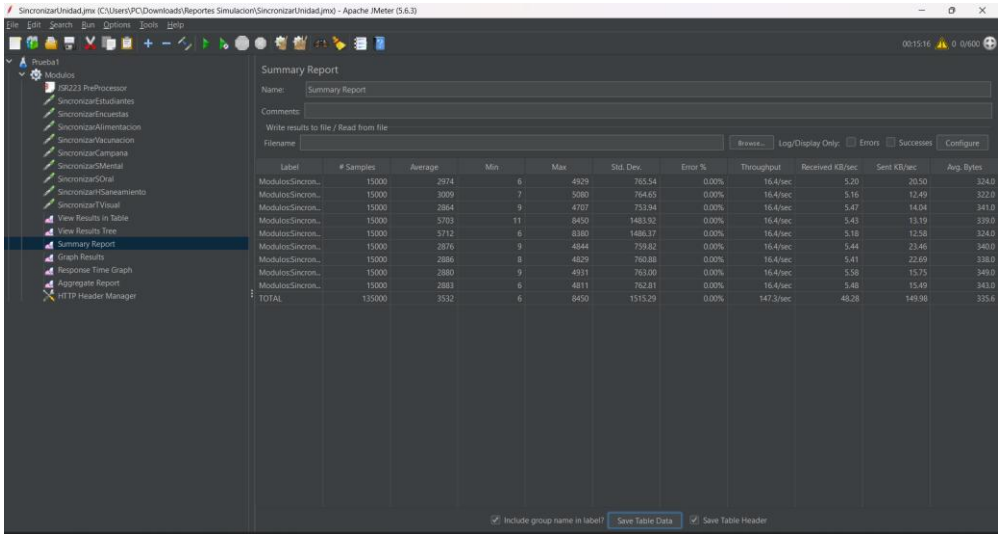
Campo	Tipo de Dato	Descripción
id	uuid	Identificador del registro.
encuesta_id	uuid	ID de encuesta (FK).
participa_riesgos_vectores	boolean	Participa en control de vectores.
participa_espacios_saludables	boolean	Participa en espacios saludables.
participa_lavado_manos	boolean	Participa en lavado de manos.
participa_agua_segura	boolean	Participa en agua segura.
numero_vectores	integer	Numero de sesiones sobre vectores.
numero_espacios_saludables	integer	Numero de sesiones de espacios saludables.
numero_lavado_manos	integer	Numero de sesiones de lavado de manos.
numero_agua_segura	integer	Numeros de sesiones de agua segura.

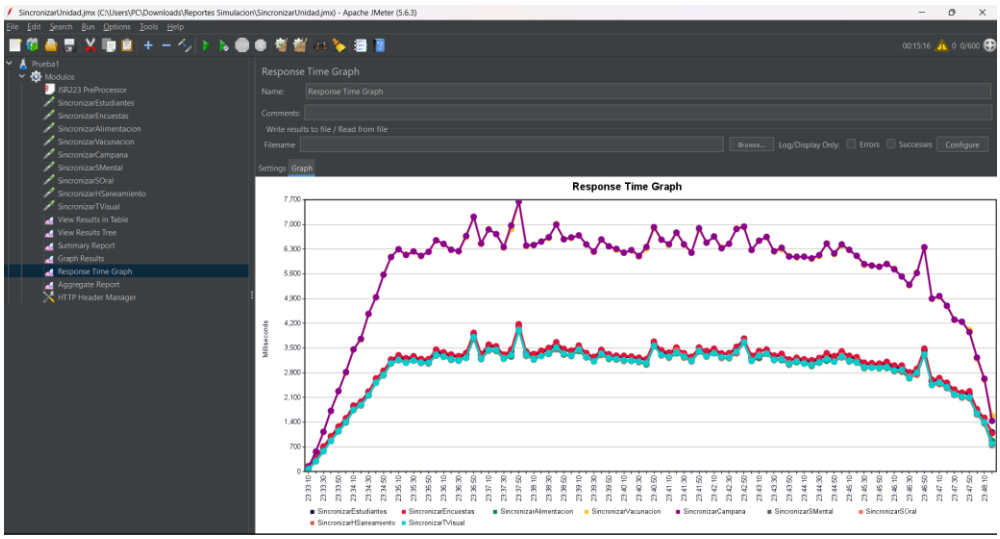
Anexo II: Pruebas de simulaciones

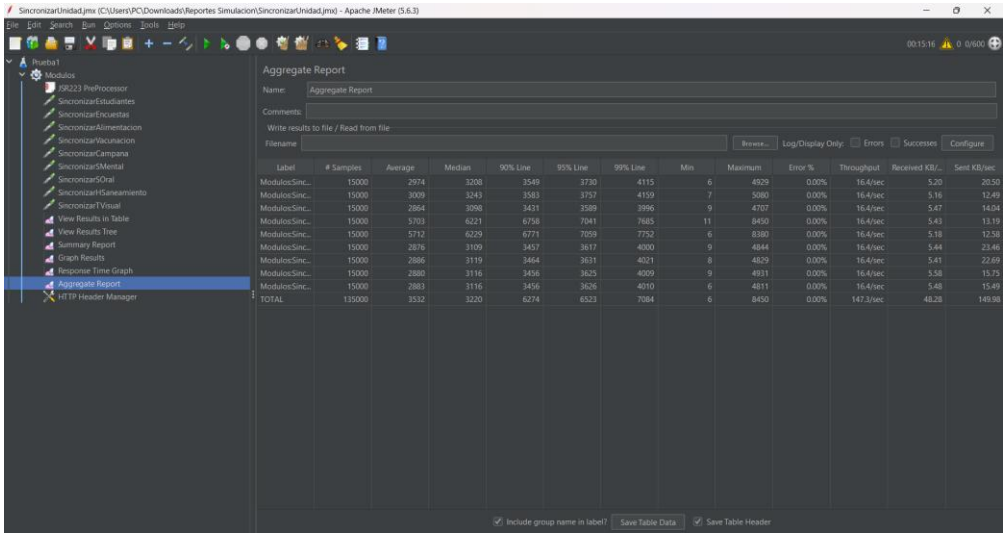
a) Eficiencia de datos


SincronizarUnidad.jmx (C:\Users\PC\Downloads\Reportes Simulacion\SincronizarUnidad.jmx) - Apache JMeter (5.6.3)
View Results in Table

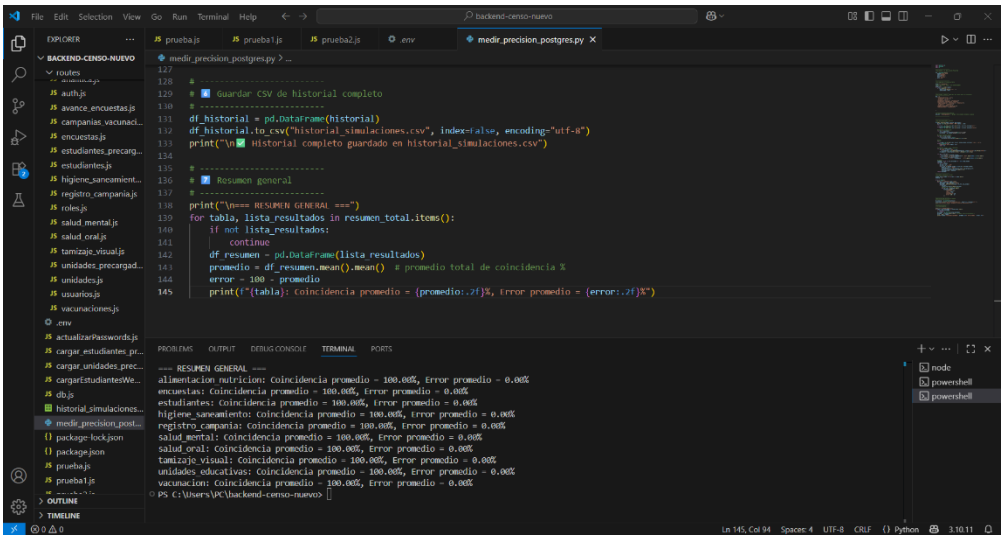

SincronizarUnidad.jmx (C:\Users\PC\Downloads\Reportes Simulacion\SincronizarUnidad.jmx) - Apache JMeter (5.6.3)
View Results Tree

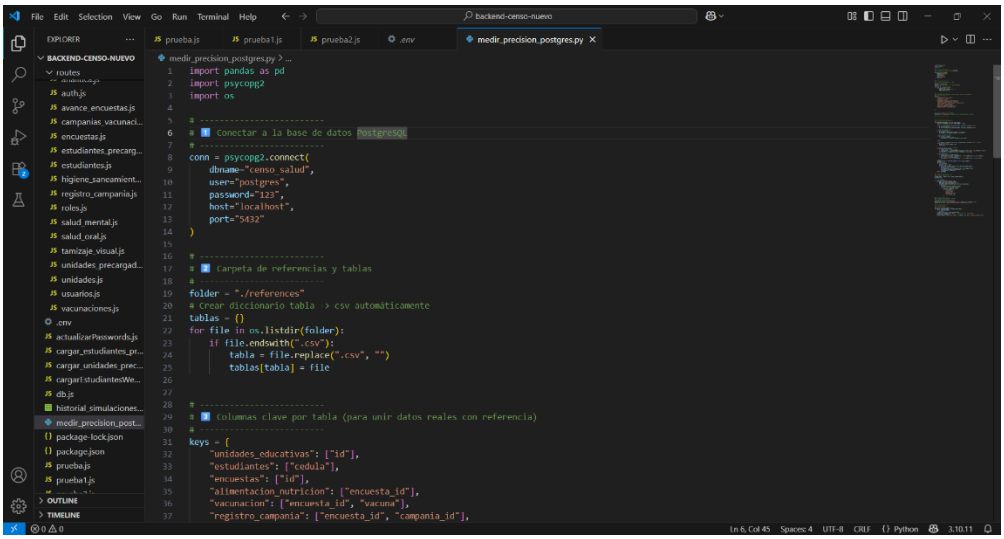

SincronizarUnidad.jmx (C:\Users\PC\Downloads\Reportes Simulacion\SincronizarUnidad.jmx) - Apache JMeter (5.6.3)
Summary Report



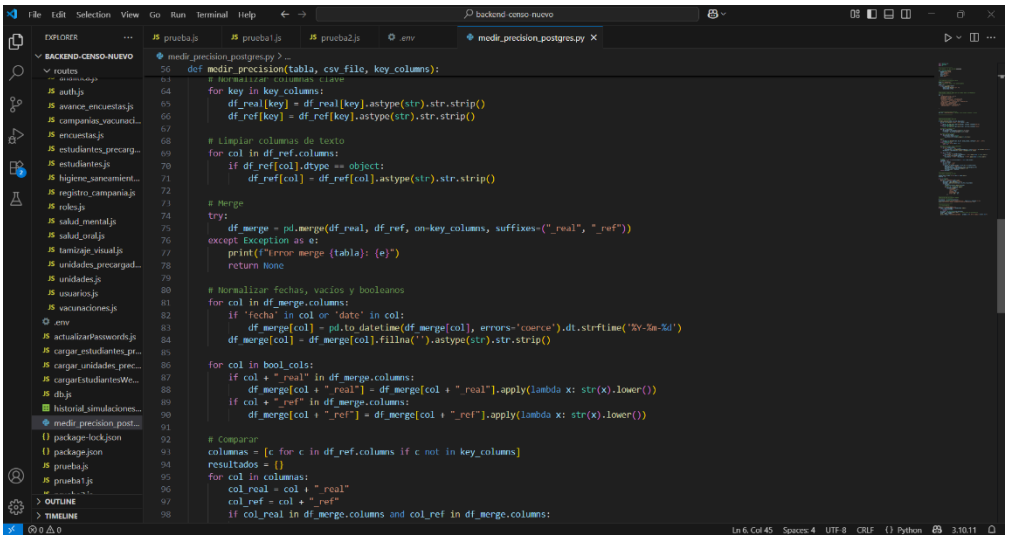


b) Precisión de los datos

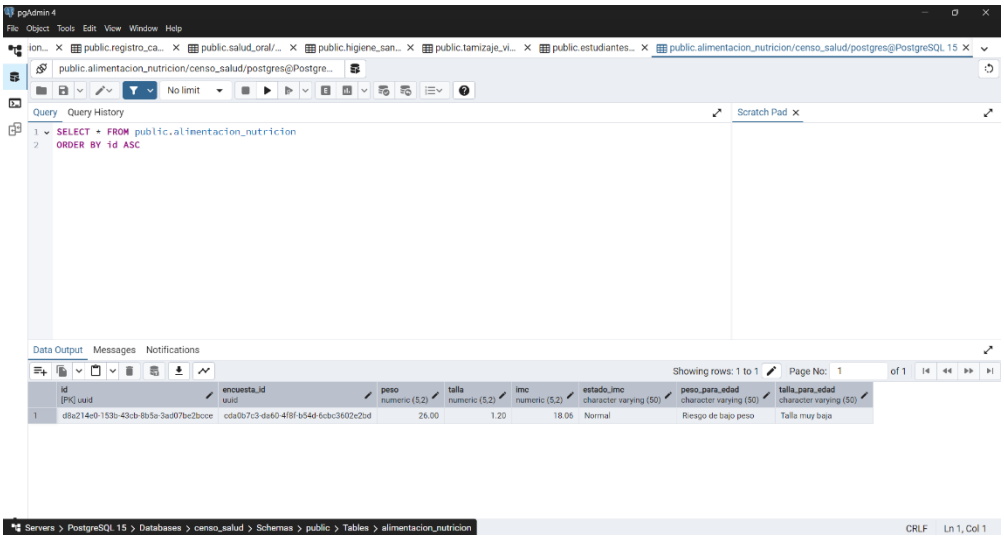




```python
import pandas as pd
import psycopg2
import os

# -------------------------
# 🔌 Conectar a la base de datos PostgreSQL
# -------------------------
conn = psycopg2.connect(
    dbname="censo_salud",
    user="postgres",
    password="123",
    host="localhost",
    port="5432"
)

# -------------------------
# 📂 Carpeta de referencias y tablas
# -------------------------
folder = "./references"
# Crear diccionario tabla -> csv automáticamente
tablas = {}
for file in os.listdir(folder):
    if file.endswith(".csv"):
        tabla = file.replace(".csv", "")
        tablas[tabla] = file


# -------------------------
# 🔑 Columnas clave por tabla (para unir datos reales con referencia)
# -------------------------
keys = {
    "unidades_educativas": ["id"],
    "estudiantes": ["cedula"],
    "encuestas": ["id"],
    "alimentacion_nutricion": ["encuesta_id"],
    "vacunacion": ["encuesta_id", "vacuna"],
    "registro_campania": ["encuesta_id", "campania_id"],
```



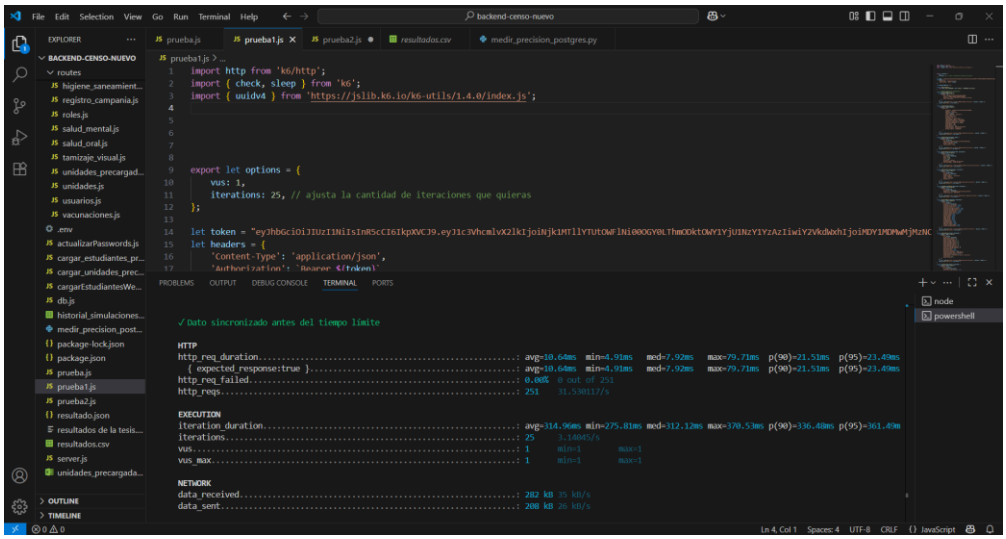
```python
def medir_precision(tabla, csv_file, key_columns):
    # Normalizar columnas clave
    for key in key_columns:
        df_real[key] = df_real[key].astype(str).str.strip()
        df_ref[key] = df_ref[key].astype(str).str.strip()

    # Limpiar columnas de texto
    for col in df_ref.columns:
        if df_ref[col].dtype == object:
            df_ref[col] = df_ref[col].astype(str).str.strip()

    # Merge
    try:
        df_merge = pd.merge(df_real, df_ref, on=key_columns, suffixes=("_real", "_ref"))
    except Exception as e:
        print(f"Error merge {tabla}: {e}")
        return None

    # Normalizar fechas, vacíos y booleanos
    for col in df_merge.columns:
        if 'fecha' in col or 'date' in col:
            df_merge[col] = pd.to_datetime(df_merge[col], errors='coerce').dt.strftime('%Y-%m-%d')
        df_merge[col] = df_merge[col].fillna('').astype(str).str.strip()

    for col in bool_cols:
        if col + "_real" in df_merge.columns:
            df_merge[col + "_real"] = df_merge[col + "_real"].apply(lambda x: str(x).lower())
        if col + "_ref" in df_merge.columns:
            df_merge[col + "_ref"] = df_merge[col + "_ref"].apply(lambda x: str(x).lower())

    # Comparar
    columnas = [c for c in df_ref.columns if c not in key_columns]
    resultados = {}
    for col in columnas:
        col_real = col + "_real"
        col_ref = col + "_ref"
        if col_real in df_merge.columns and col_ref in df_merge.columns:
```

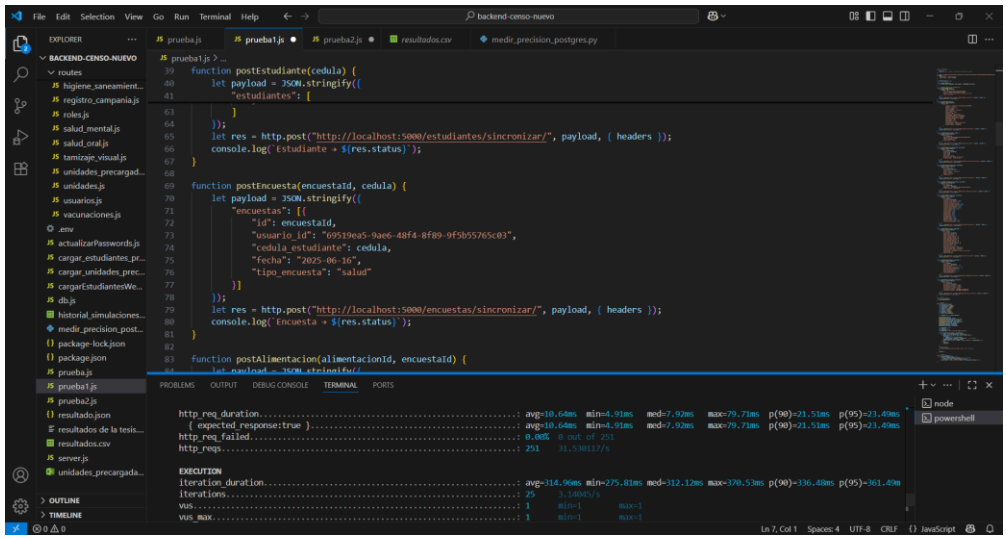


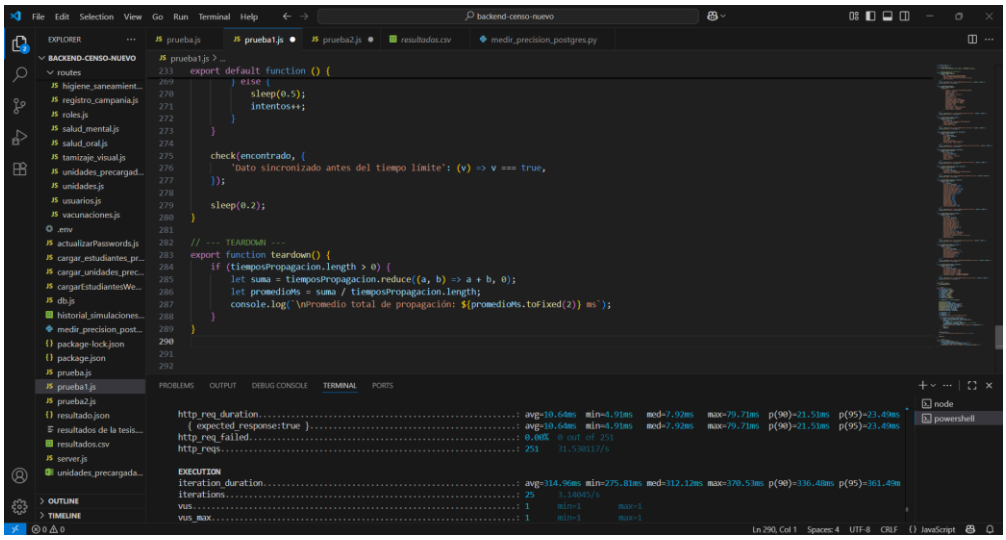
```
SELECT * FROM public.alimentacion_nutricion
ORDER BY id ASC
```

id [PK] uuid	encuesta_id uuid	peso numeric (5,2)	talla numeric (5,2)	imc numeric (5,2)	estado_imc character varying (50)	peso_para_edad character varying (50)	talla_para_edad character varying (50)
d8a214e0-153b-43cb-8b5e-3ad07be2bcce	cda0b7c3-da60-4f8f-b54d-6cbc3602e2bd	26.00	1.20	18.06	Normal	Riesgo de bajo peso	Talla muy baja

c) Disponibilidad







Anexo III: Ficha técnica del sistema

Ficha técnica del sistema de gestión de censo de salud Chambo	
Indicador/Característica	**Valor/Descripción**
Nombre del sistema	Censo de salud Chambo.
Tipo de aplicación	Aplicación web y Aplicación móvil.
Tamaño de la aplicación (APK)	43.58 MB.
Disponibilidad del sistema	Android (versión mínima 8.0) y sistema web compatible con navegadores modernos.
Módulos implementados	10 módulos: Unidad educativa, estudiantes, encuestas, alimentación, vacunación, campañas de vacunación, salud mental, salud oral, higiene y saneamiento, tamizaje visual.
Tiempo de desarrollo	4 meses.
Tecnologías utilizadas	Flutter, Node.js, React.js.
Metodología de desarrollo	Mobile-D.
Bases de datos	MySQL, PostgreSQL.
Modo Offline	Si. Permite trabajo sin conexión y posterior sincronización.

Anexo IV: Manual de usuario

MANUAL DE USUARIO

SISTEMA PARA LA GESTIÓN DE CENSO DE SALUD EN EL CENTRO DE SALUD CHAMBO UTILIZANDO EL FRAMEWORK FLUTTER

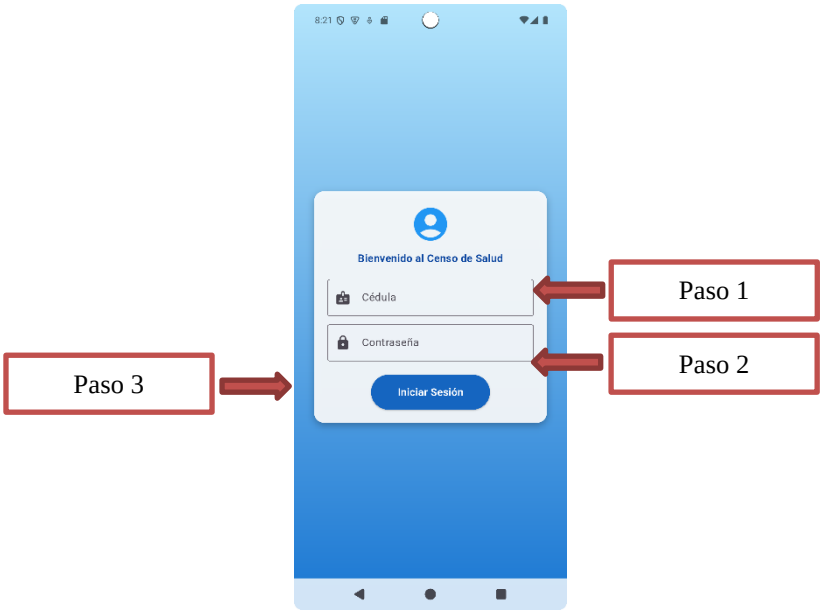
CENTRO DE SALUD CHAMBO

Tabla de contenido

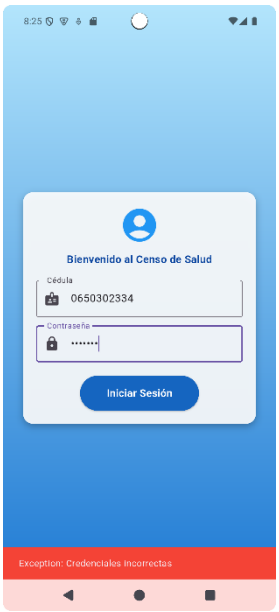
1. Apartado móvil

1.1 Inicio de sesión

Al iniciar sesión por primera vez a través de la cedula y contraseña, la aplicación descargara el listado de los estudiantes, unidades educativas y campañas de vacunación pre cargadas. Esta sincronización inicial puede tardar unos minutos, dependiendo de la cantidad de estudiantes y la calidad de la conexión. El usuario deberá esperar a que se complete antes de empezar con el registro de encuestas.



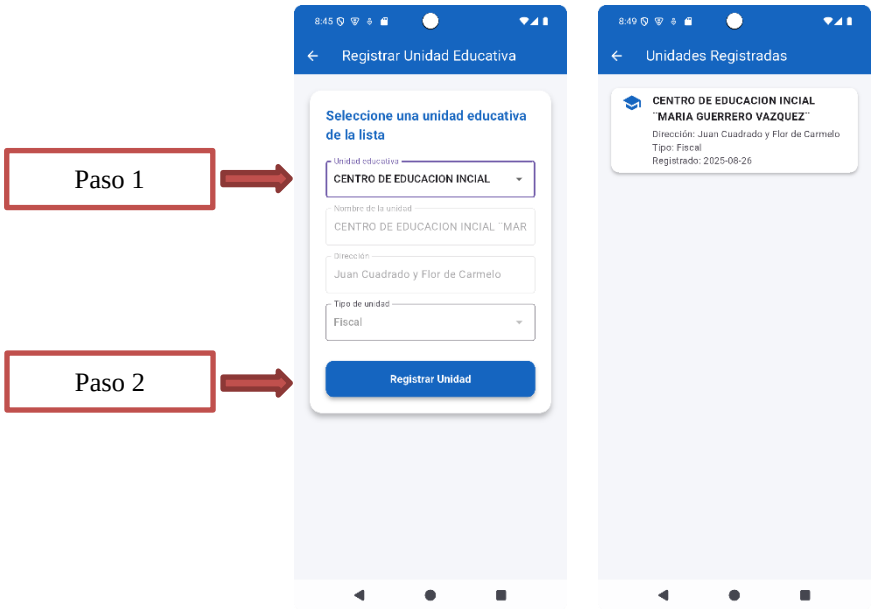
En caso de que las credenciales sean incorrectas, se mostrará el siguiente mensaje de advertencia.



Una vez iniciada la sesión, el sistema mostrará la pantalla principal, la cual dispone de un menú lateral que organiza los distintos módulos: selección de unidad educativa,

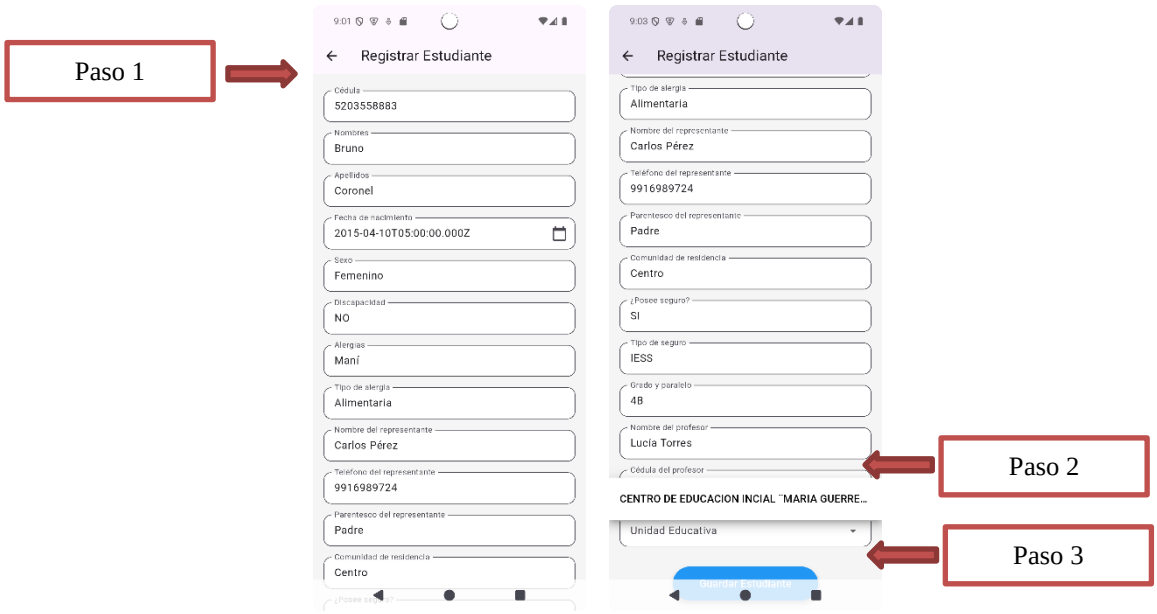
visualización de unidades registradas, selección de estudiantes, registro de encuestas de salud, consulta de encuestas registradas, consulta de estudiantes registrados y sincronización de información.

En esta sección, el encuestador deberá seleccionar la unidad educativa y presionar el botón **registrar unidad** para dar inicio al proceso de registro. Posteriormente, mediante la opción **ver unidades registradas**, será posible verificar si la unidad educativa fue seleccionada correctamente antes de continuar con el proceso.

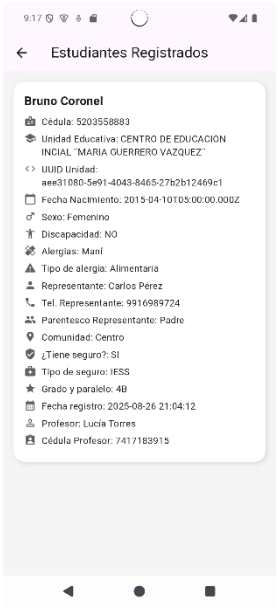


Al utilizar **seleccionar estudiante**, el encuestador podrá ingresar el número de cédula del estudiante, lo que permitirá que el sistema complete automáticamente las casillas con la información registrada previamente. En esta etapa, el encuestador deberá confirmar la

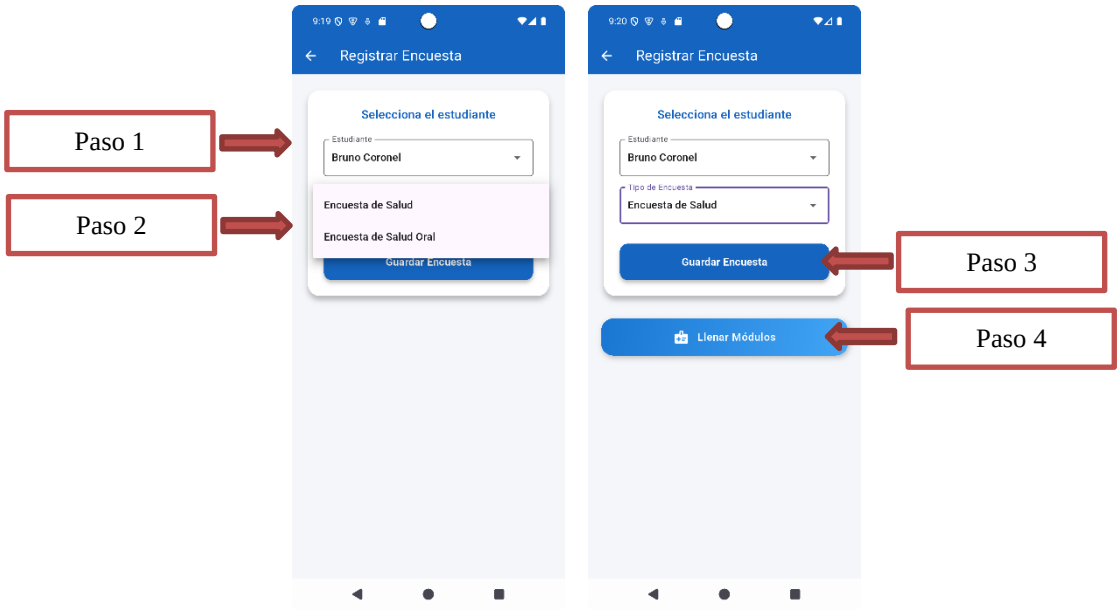
unidad educativa a la que pertenece y posteriormente, presionar el botón **guardar estudiante** para asociar y registrar correctamente los datos en el sistema.



A través de la opción **ver estudiantes registrados**, el encuestador podrá visualizar al estudiante previamente registrado junto con toda su información asociada, lo que permite verificar la correcta incorporación de los datos en el sistema antes de continuar con el levantamiento de la encuesta.



Mediante **llenar encuesta de salud**, se podrá seleccionar al estudiante correspondiente y definir el tipo de encuesta que desea realizar. Una vez completada esta selección, será necesario presionar el botón **guardar encuesta**, lo que habilitará automáticamente la opción **llenar módulos.** A través de esta función se activarán los diferentes módulos de la encuesta de salud.



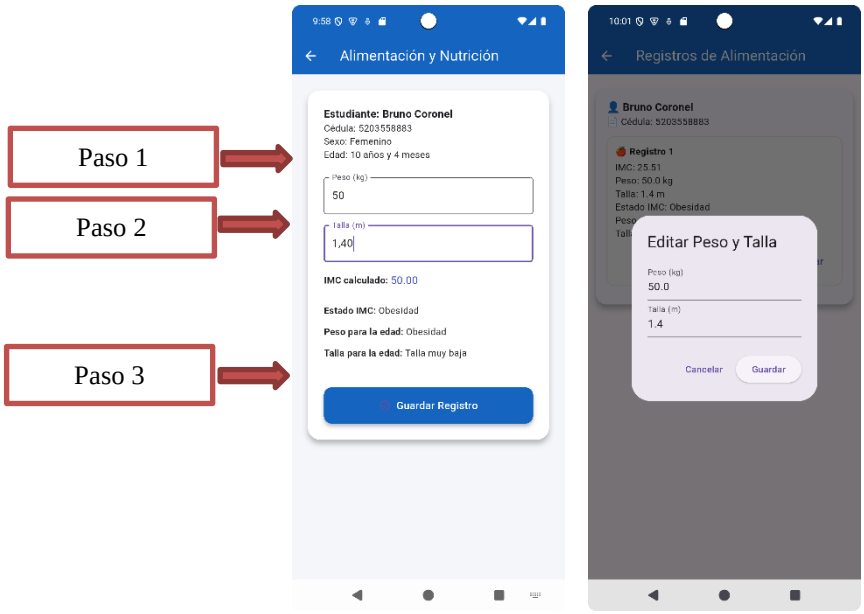
1.2 Encuesta de salud

En la pantalla, se muestra en la parte superior el nombre del estudiante con su número de cédula. Desde esta sección se podrá accede a los diferentes módulos disponibles: Alimentación y nutrición, Vacunación, Tamizaje visual, Salud mental e Higiene y saneamiento, cada uno con su respectiva opción para registrar información y consultar registros anteriores.

1.2.1 Módulo de alimentación y nutrición

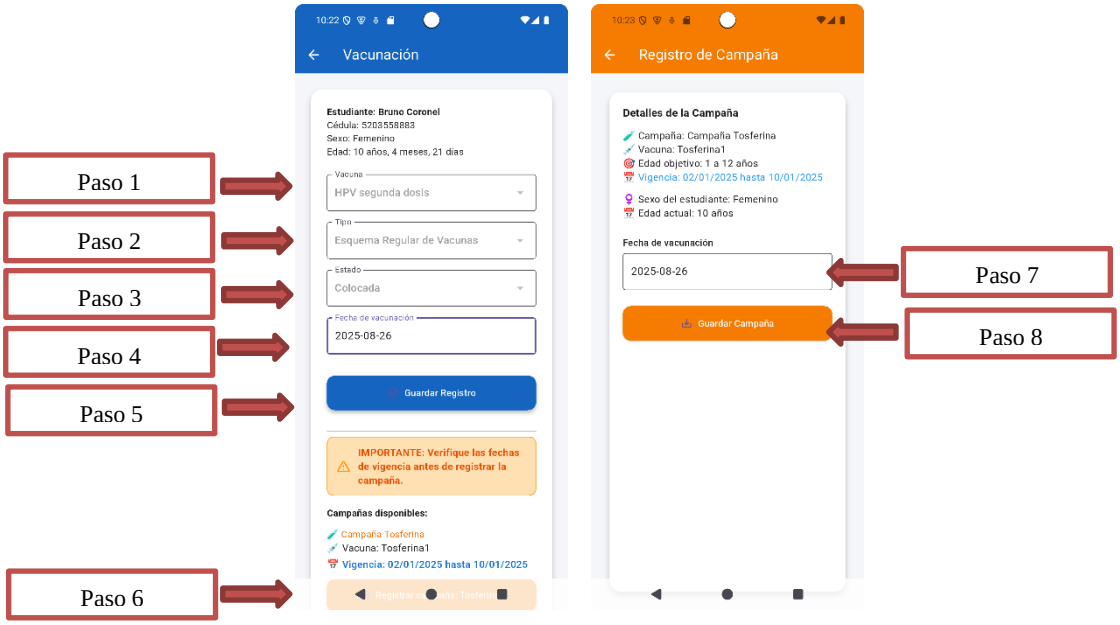
En el módulo de **alimentación y nutrición** se presentan los datos del estudiante, incluyendo nombre, cédula, sexo y edad. Desde esta sección el encuestador podrá ingresar el peso (kg) y la talla (m), a partir de los cuales el sistema calculara automáticamente el IMC, posteriormente al seleccionar **guardar registros** la información se almacena de forma segura

y mediante el botón **ver registro de alimentación** se podrá consultara y editar el peso y la talla del encuestado.



1.2.2 Módulo de vacunación

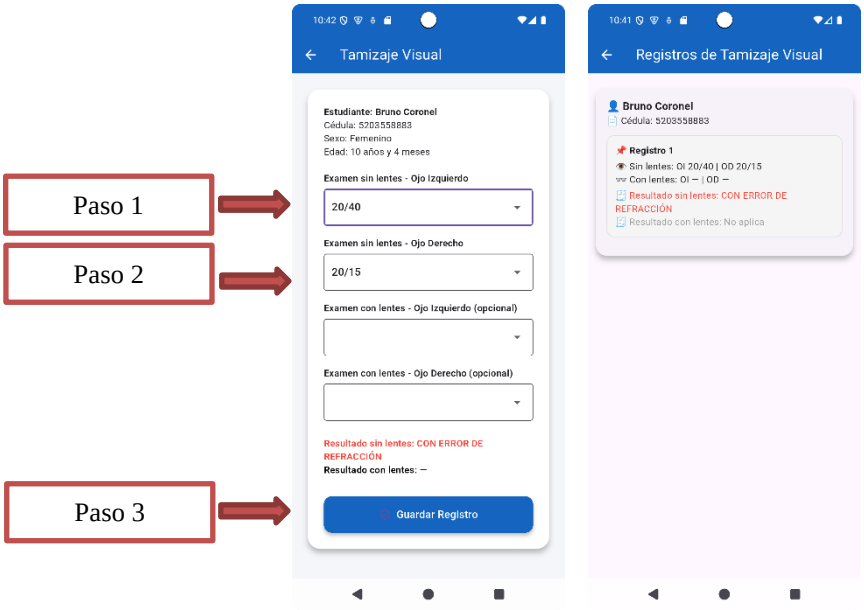
En el módulo de **vacunación** se podrá seleccionar el nombre de la vacuna, el tipo, el estado y la fecha de aplicación. Al presionar el botón **guardar registro**, la información se almacenará de manera segura. Además, dependiendo de la edad del estudiante, se habilitará la campaña correspondiente, en la cual únicamente será necesario seleccionar la fecha y guardar la campaña.



En **registro de vacunación** se podrá visualizar de manera completa toda la información registrada, incluyendo los datos de la vacuna, tipo, estado, fecha de aplicación y campañas asociadas, lo que permite un seguimiento preciso y actualizado del historial de vacunación del estudiante.

1.2.3 Módulo de tamizaje visual

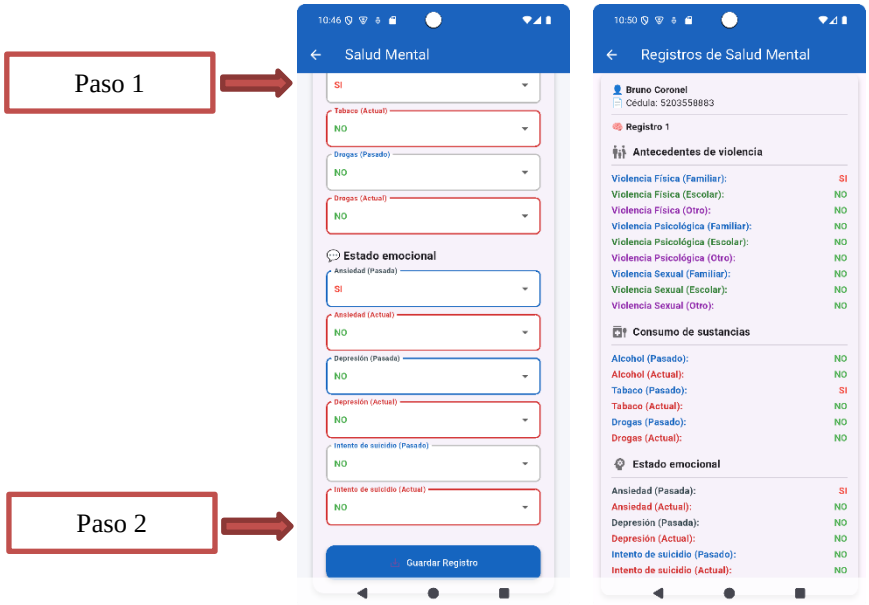
En el módulo de **tamizaje visual** se podrá registrar el puntaje de refracción del ojo izquierdo y del ojo derecho, tanto sin lentes como con lentes. A partir de estos datos, el sistema determinará si existe algún error de refracción. Posteriormente, la información se almacenará mediante el botón **guardar registro**. Por otra parte, en la sección **ver registro de tamizaje visual** se podrá consultar toda la información generada.



1.2.4 Módulo de salud mental

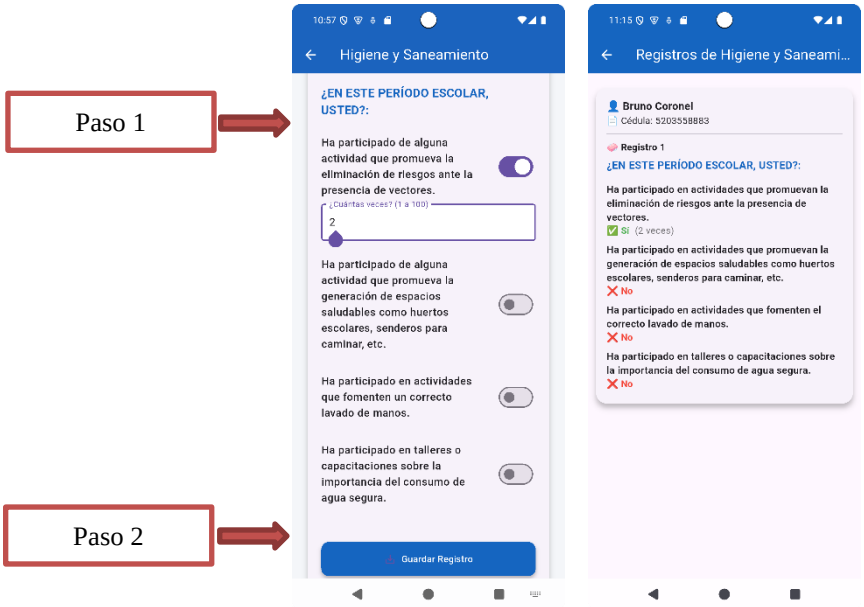
En el módulo de **salud mental** se presentarán preguntas cerradas de **SI** o **NO** relacionadas con antecedentes de violencia, consumo de sustancias y estado emocional del estudiante. La información se almacenará al presionar el botón **guardar información**. Además, en la sección **ver registro de salud mental** se podrá consultar toda la información registrada en

el módulo, lo que permite un seguimiento preciso del estado emocional y conductual del estudiante.



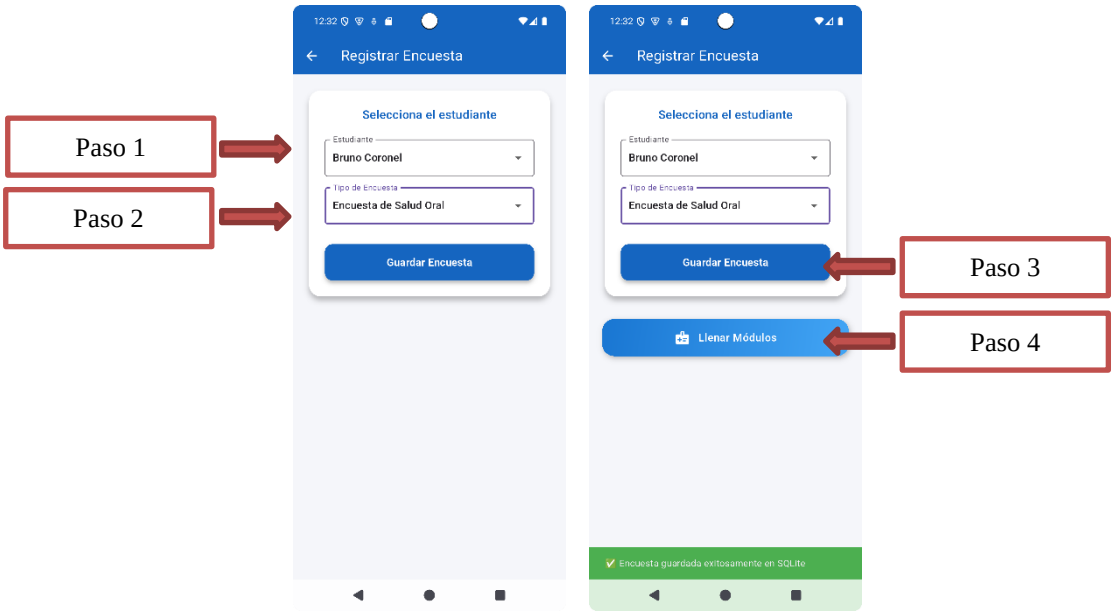
1.2.5 Módulo de higiene y saneamiento

En el módulo de **higiene y saneamiento** se presentará una serie de preguntas relacionadas con las actividades realizadas por el estudiante durante el periodo escolar, en las cuales se deberá activar la casilla correspondiente y registrar el número de veces que se han realizado. La información se almacenará al presionar el botón **guardar información**. Además, en la sección **ver registro de higiene** se podrá consultar toda la información almacenada, lo que permite un seguimiento detallado de los hábitos del estudiante.

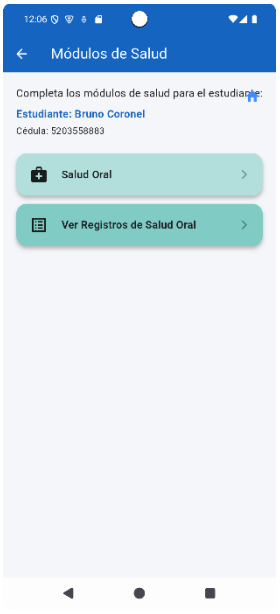


1.3 Encuesta de salud oral

En la sección de registro, el encuestador con rol de odontólogo seleccionará al estudiante y la opción **encuesta de salud oral**, posteriormente deberá presionar el botón **guardar encuesta**, lo que habilitará el botón **llenar módulos**; al presionar este último, se desplegará el módulo correspondiente, permitiendo registrar de manera organizada y completa todos los datos de la encuesta.



En esta sección se podrá visualizar el nombre del estudiante, su cédula, el módulo de salud oral y la opción ver registro de salud oral, lo que permite consultar de manera organizada y completa toda la información registrada en el módulo.

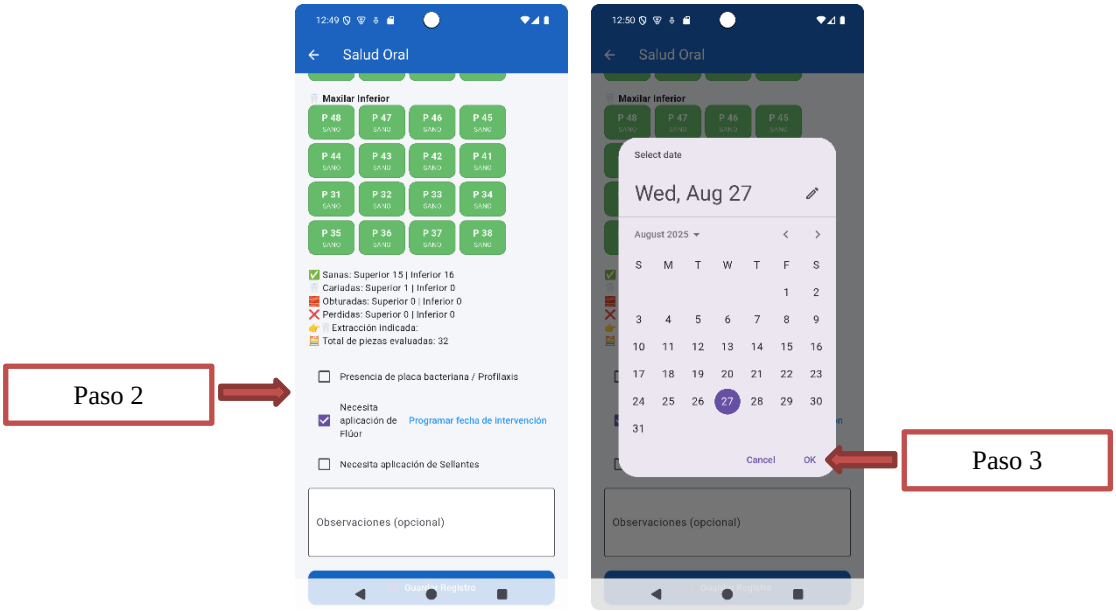


1.3.1 Módulo de salud oral

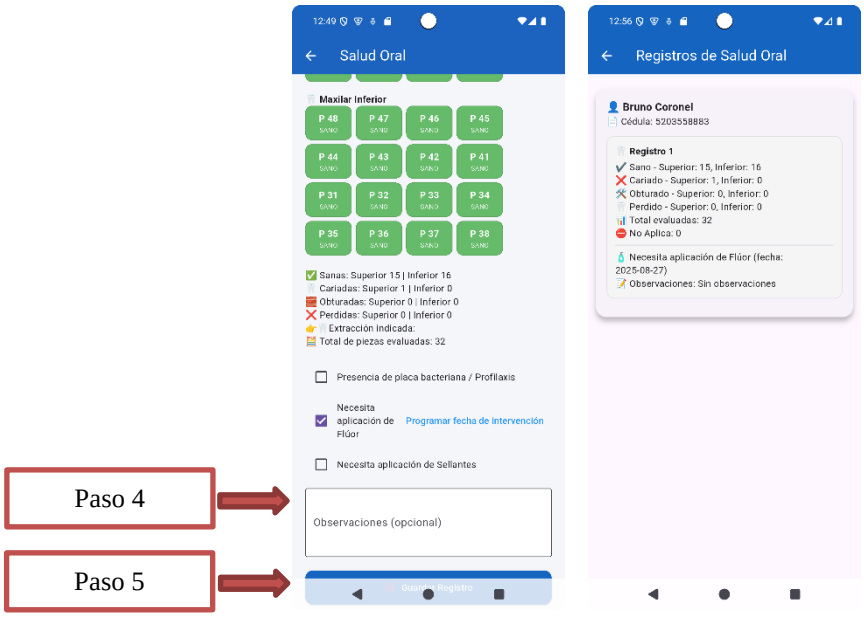
En el módulo de **salud oral** se pueden observar las piezas dentales del maxilar superior e inferior, donde el encuestador podrá seleccionar cada pieza dental y asignarle el estado correspondiente entre las opciones: sano**,** cariado**,** obturado**,** perdido**,** extracción indicada o no aplica. A partir de estas selecciones, el sistema mostrará automáticamente el número de

piezas sanas**,** obturadas**,** perdidas y con extracción indicada, así como el total de piezas evaluadas.

Adicionalmente, se presentan tres casillas referentes a: presencia de placa bacteriana**,** necesita aplicación de flúor y necesita aplicación de sellante; al seleccionar alguna de estas opciones, se habilita el botón **programar fecha de intervención**, que permite registrar la fecha prevista para la intervención.



El módulo también cuenta con un cuadro de observaciones para registrar notas adicionales. Para almacenar toda la información registrada, el encuestador deberá presionar el botón **guardar registro**, garantizando que los datos queden completos y actualizados. Asimismo, en el módulo **ver registro de salud oral** se podrá visualizar toda la información recolectada, permitiendo consultar de manera organizada los estados de las piezas dentales, intervenciones programadas y observaciones registradas, lo que facilita un seguimiento preciso del estado de salud oral del estudiante.

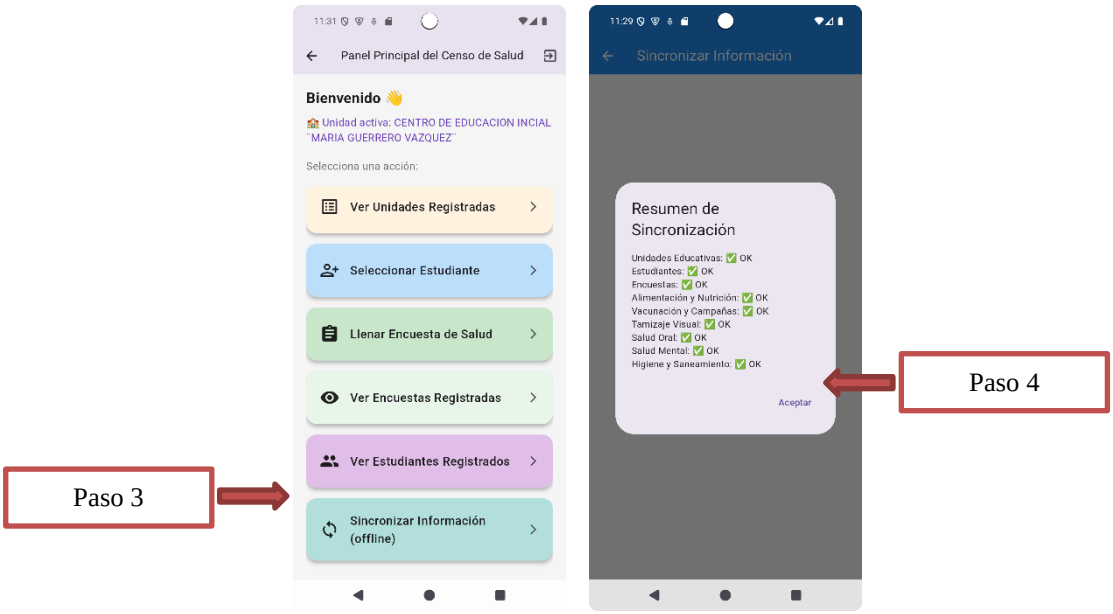


1.4 Sincronizar datos

Al finalizar los módulos, el encuestador podrá retroceder presionando este icono ← ubicado en la parte superior izquierda. Al hacerlo, se mostrará un mensaje que confirma los cambios realizados en la encuesta actual y pregunta si se desea regresar al menú principal, debiendo seleccionar la opción **sí** para regresar.

En esta sección, el encuestador de la área de salud y odontología deberá contar con conexión a internet para sincronizar la información que se ha almacenado de manera offline en el dispositivo móvil, asegurando que todos los datos recopilados queden actualizados en la base de datos central, lo cual se podrá realizar mediante la selección del botón **sincronizar información**, que inicia el proceso de transferencia de datos de forma segura y confiable,

mostrando a continuación un cuadro de resumen de sincronización en el que el encuestador deberá presionar el botón **aceptar** para confirmar y cerrar el mensaje.



1.5 Cerrar sesión

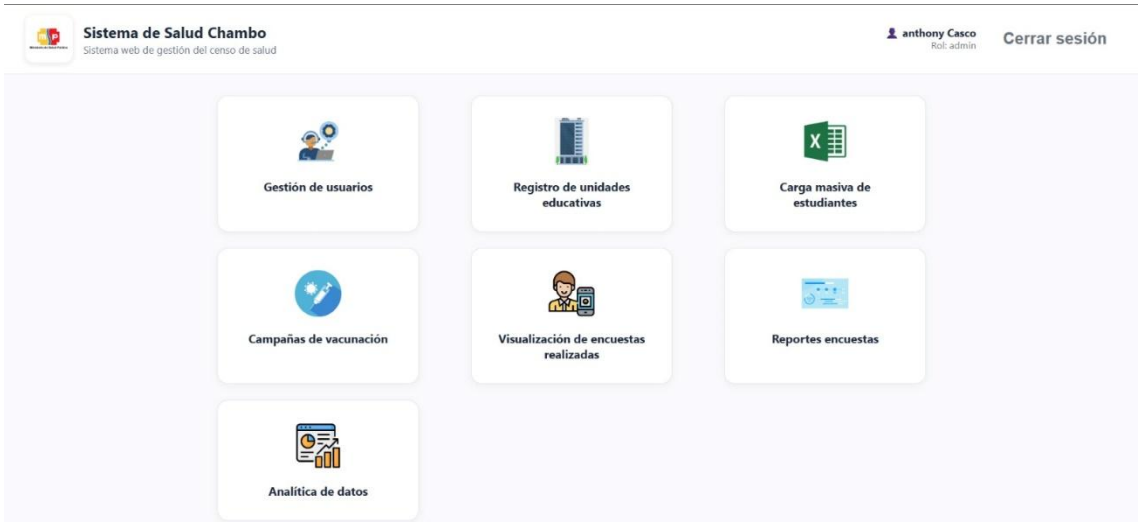
Para **cerrar sesión**, el usuario deberá hacer clic en el icono de ⍈, en el caso de requerir un nuevo acceso será necesario autenticarse nuevamente con sus credenciales.

2. Apartado web

2.1 **2.1 Inicio de sesión**

En la **ventana principal**, el usuario deberá ingresar su **cédula** y **contraseña**, tal como se indica en la ilustración a continuación.

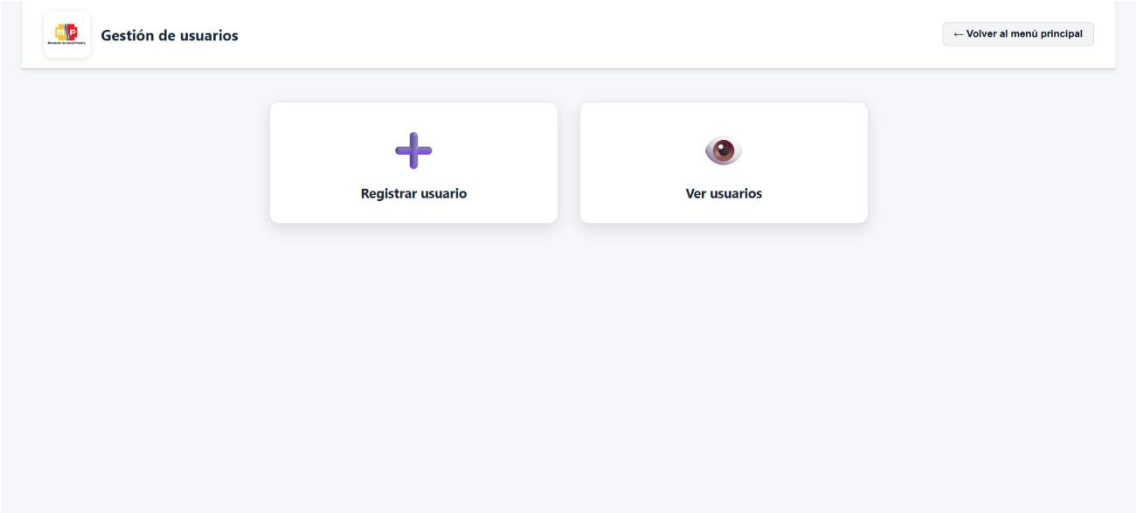
En esta sección se encuentra el módulo principal, que integra las opciones de gestión de usuarios, analítica de datos, campañas de vacunación, unidades educativas, carga masiva de estudiantes, avance de encuestas y ver estudiantes.



2.2 Módulos del sistema

2.2.1 Gestión de usuarios

En esta sección, el usuario con rol de **administrador** podrá acceder a las opciones de registrar un nuevo usuario y ver los usuarios registrados**.**

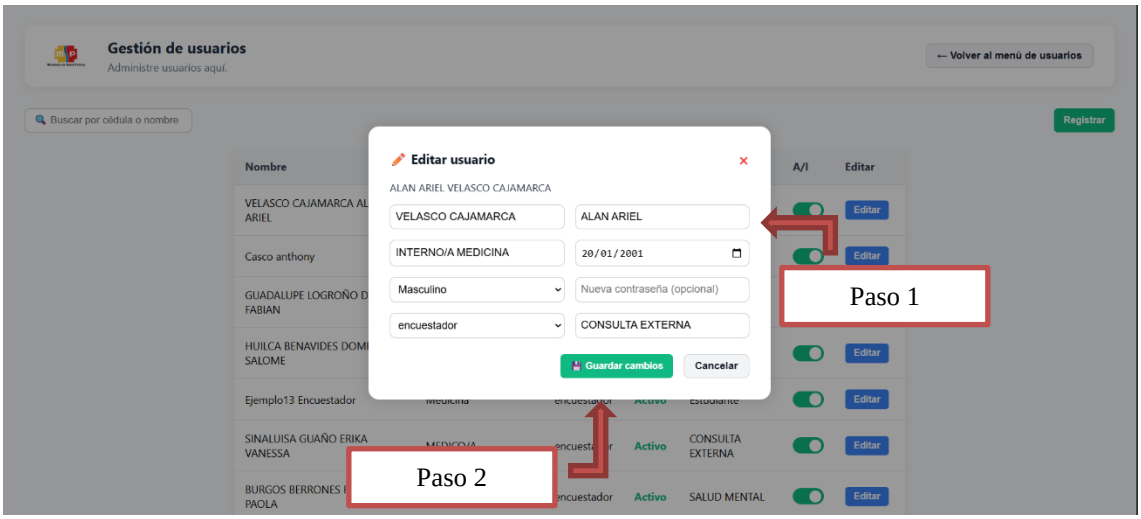


Al seleccionar la opción **registrar usuario**, se podrá crear un perfil ingresando los datos correspondientes: número de cédula, nombres, apellidos, carrera, fecha de nacimiento, sexo, contraseña, rol y área de trabajo. Finalmente, para guardar el registro, se deberá presionar el botón **aceptar**.

En el módulo de **gestión de usuarios**, el administrador podrá buscar a los usuarios por **cédula** o **nombre** utilizando el **icono de la lupa**. Además, tendrá la opción de **editar**, **activar o desactivar** usuarios según sea necesario y contará con un **botón de acceso rápido** para registrar un nuevo usuario de manera inmediata.

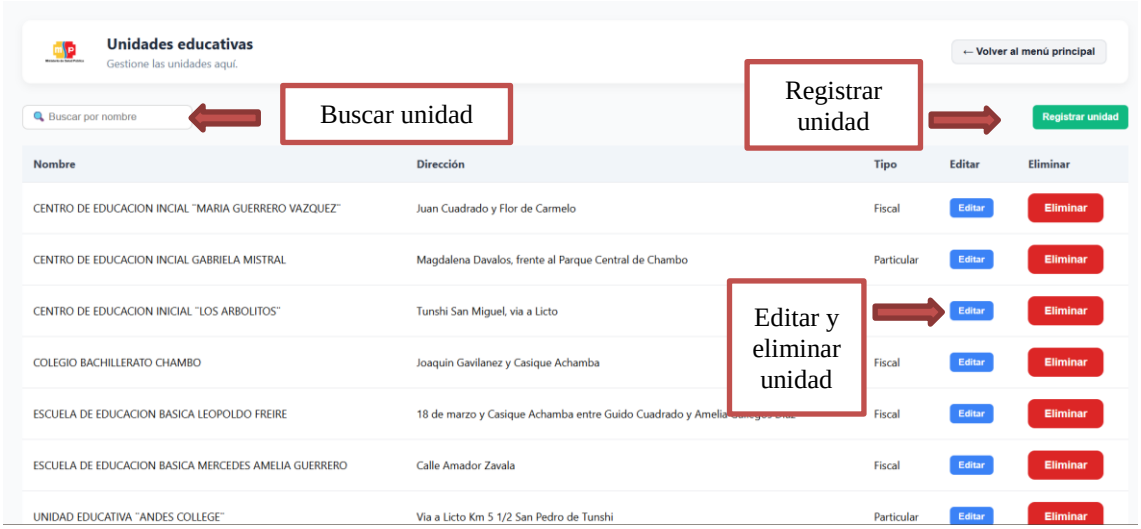
Si el administrador selecciona la opción **editar**, se mostrará una ventana donde podrá modificar todos los campos de información del usuario previamente registrado y, al presionar el botón **guardar cambios**, se actualizará la información en el sistema.

Por otra parte, solo el administrador tiene la facultad de gestionar las contraseñas de los usuarios.



2.2.2 Registro de unidades educativas

En esta sección, el administrador podrá visualizar las unidades educativas y contará con las opciones de **buscar**, **editar**, **eliminar** y **registrar** nuevas unidades según sea necesario.



Al seleccionar la opción **registrar unidad**, se deberán completar los campos de **nombre**, **tipo** y **dirección**, y posteriormente presionar el botón **guardar** para almacenar la información.

Al seleccionar **editar**, se podrá modificar la información registrada y presionar **guardar** para actualizar los datos en el sistema.

2.2.3 Carga masiva de estudiantes

El usuario deberá aplastar el botón **elegir archivo** para seleccionar el Excel con la lista de estudiantes; luego, al presionar **carga masiva de datos**, el sistema valida la información, guarda los registros correctos y muestra un reporte con los cargados y rechazados.

Al seleccionar el botón **eliminar todos los estudiantes precargados**, el usuario borrará de forma permanente la información cargada masivamente. Al ejecutar esta acción, se mostrará una ventana para **confirmar o cancelar** la operación. Esta acción debe realizarse únicamente en casos necesarios, cuando se requiera **reiniciar la base de datos** para cargar un nuevo listado actualizado de estudiantes.

2.2.4 Campañas de vacunación

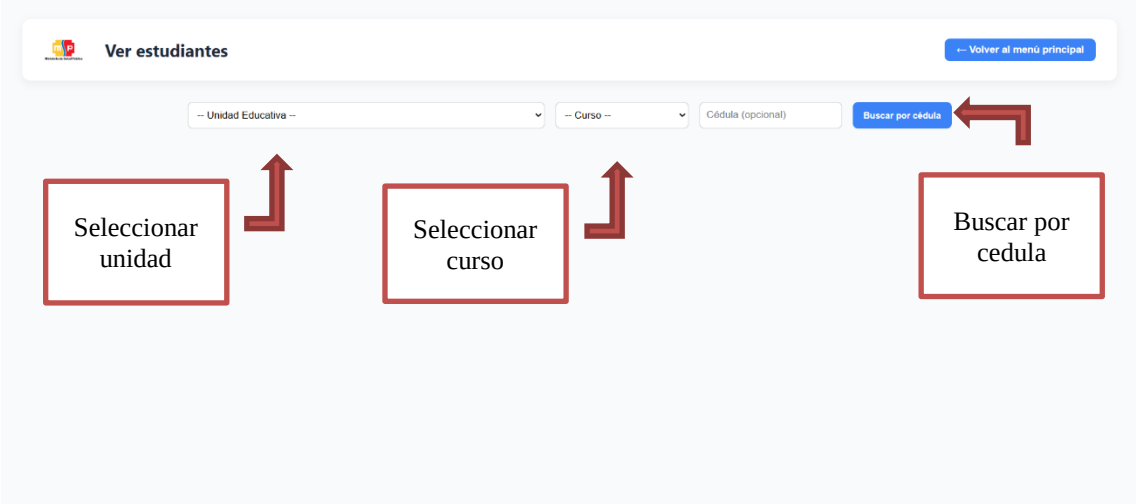
En esta sección, el usuario podrá visualizar la información de las campañas de vacunación, incluyendo **nombre, vacuna, rango de edad** y **rango de fechas**. Además, cuenta con las funciones de **editar campaña**, **registrar una nueva campaña** y **buscar campaña por nombre**, facilitando la gestión y actualización de los registros.

Al presionar **editar**, se mostrará una ventana en la que el usuario podrá modificar el **nombre de la campaña**, **vacuna**, **rango de edad** y el **rango de fechas**. Para aplicar los cambios, será necesario presionar el botón **guardar**; en caso de no querer actualizar la información, solo se deberá presionar el **icono de X** para cerrar la ventana sin guardar los cambios.

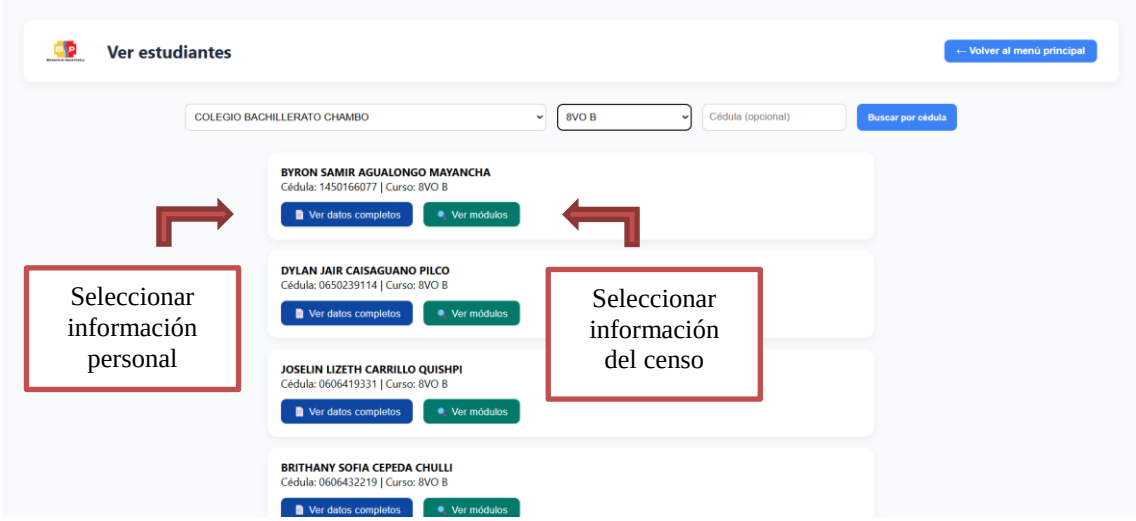
Al presionar **registrar nueva campaña**, el usuario podrá completar todos los campos de información correspondientes a la nueva campaña y, posteriormente, presionar el botón **guardar** para registrar la campaña en el sistema o, en caso de no querer guardar, presionar el **icono de X** para cerrar la ventana sin realizar cambios.

2.2.5 Visualización de encuestas realizadas

En **encuestas realizadas**, el usuario podrá filtrar la información por **unidad educativa**, **curso** o buscar directamente por el **número de cédula** del estudiante, facilitando la localización rápida de los registros.

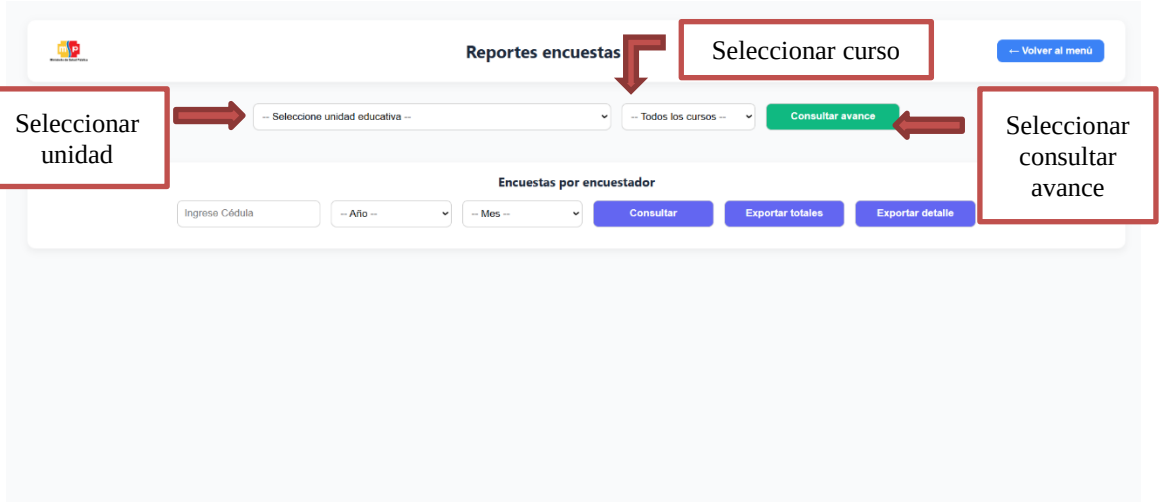


Al filtrar al estudiante, el usuario dispondrá de dos botones: uno para **visualizar la información del estudiante** y otro para **consultar el resumen del censo realizado**.

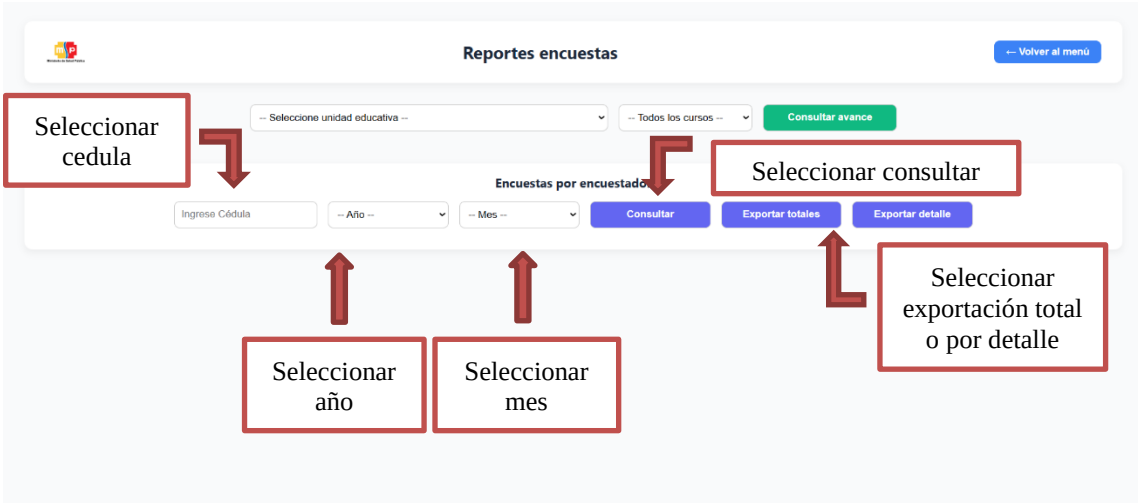


2..1 2.2.6 Reporte encuestas

El usuario podrá filtrar la información por **unidad educativa** y **curso**. Al presionar el botón **consultar avance**, el sistema mostrará los datos correspondientes, incluyendo el número de estudiantes encuestados**,** encuestas completas**,** encuestas no registradas**,** porcentaje de avance y el último estudiante encuestado.

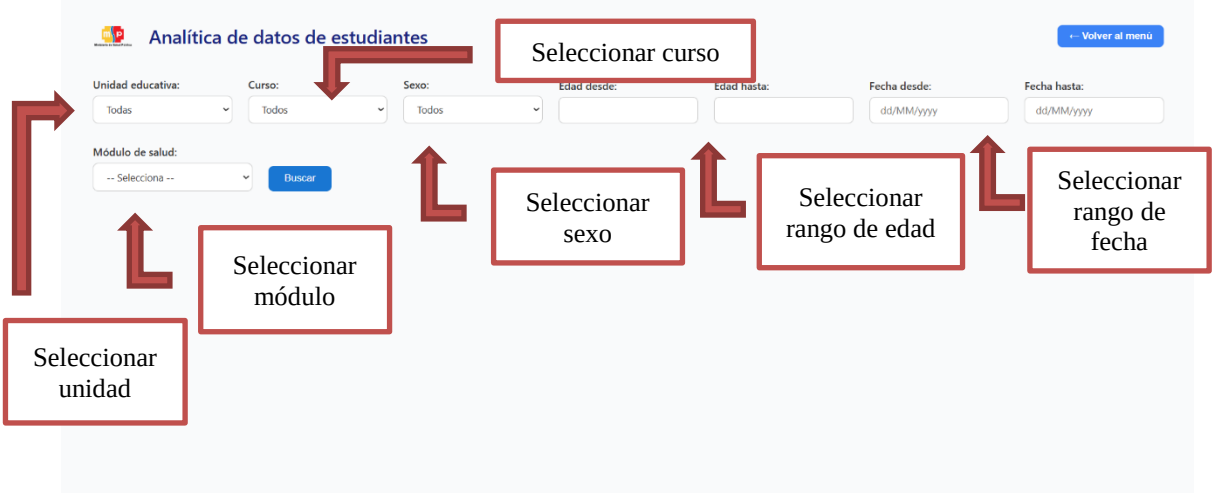


En el segundo bloque, el usuario podrá filtrar las encuestas ingresando la **cédula**, el **año** y el **mes**. Al presionar **consultar**, se mostrarán los resultados y se habilitan dos opciones de exportación: **exportar totales**, que genera un archivo con el resumen de encuestas, y **exportar detalle**, que descarga el listado completo con la información detallada de cada encuesta.

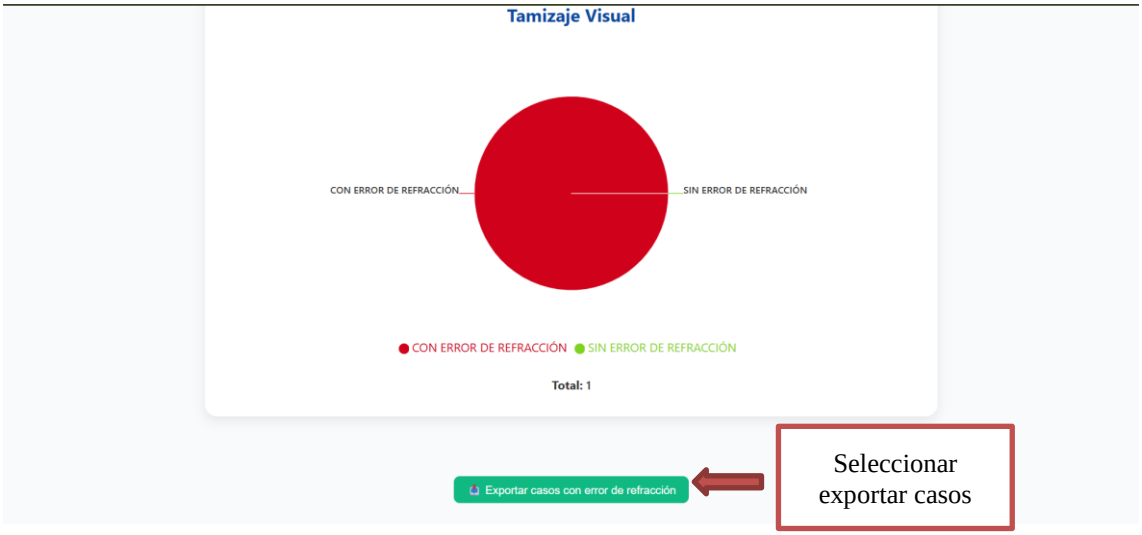


2.2.5 Analítica de datos

En esta sección, el usuario podrá filtrar la información por **unidad educativa, curso, sexo, rango de edad, rango de fechas y módulo de salud**. Para ejecutar la búsqueda, será necesario presionar el botón **buscar**.



Al filtrar la información de los censos en el módulo de analítica de datos, el usuario podrá exportar los resultados en formato xlxs para ciertos **grupos vulnerables**, presionando el botón **exportar casos**.



En este apartado, al seleccionar el módulo de **alimentación y nutrición** o **salud oral**, se habilitarán dos filtros adicionales que permiten visualizar un **histograma comparativo entre periodos**, de acuerdo con el **año** y el **mes** seleccionado.

2.3 Cerrar sesión

Para "**cerrar sesión"**, el usuario deberá hacer clic en botón **cerrar sesión**, en el caso de requerir un nuevo acceso será necesario autenticarse nuevamente con sus credenciales.

Anexo V: Manual técnico

MANUAL TÉCNICO

SISTEMA PARA LA GESTIÓN DE CENSO DE SALUD EN EL CENTRO DE SALUD CHAMBO UTILIZANDO EL FRAMEWORK FLUTTER

CENTRO DE SALUD CHAMBO

1. Introducción

El presente Manual Técnico documenta de manera detallada la estructura, configuración y funcionamiento del sistema informático desarrollado para la gestión del Censo de Salud en el Centro de Salud Chambo**.**
Este manual está dirigido a desarrolladores, técnicos y personal de soporte, y tiene como finalidad servir como guía de referencia técnica para la instalación, mantenimiento, uso y futura evolución del sistema.
En las siguientes secciones se detallan los componentes del sistema (móvil, web y backend), la arquitectura general, los requerimientos técnicos, la estructura de la base de datos, la lógica de sincronización, y los procedimientos de despliegue y administración.

2. Objetivo del sistema

El objetivo principal del sistema es digitalizar y optimizar el proceso de levantamiento y gestión de información de salud en estudiantes de unidades educativas del cantón Chambo, mediante una solución tecnológica integral compuesta por una aplicación móvil, un backend centralizado y una plataforma web administrativa.

3. Dirigido a

Este manual está orientado a desarrolladores, técnicos de soporte, y personal del área de sistemas que esté a cargo de la instalación, administración o mantenimiento del sistema de Censo de Salud.

Su lectura permitirá conocer cómo está estructurado internamente el sistema, cómo se instala, qué módulos lo componen, cómo se conecta a la base de datos y cómo se puede escalar o actualizar en el futuro.

4. Conocimientos previos

Para el uso adecuado de este manual y la correcta administración técnica del sistema informático del Censo de Salud, el personal encargado debe contar con ciertos conocimientos previos. Estos conocimientos permiten entender la estructura, configuración, despliegue y mantenimiento de los diferentes componentes del sistema.

A continuación, se detallan las áreas de conocimiento necesarias:

4.1. Conocimientos generales

Manejo básico de sistemas operativos Windows y/o Linux (instalación de programas, navegación de archivos, uso de terminal o consola).

Comprensión de conceptos de redes y conectividad: IP local, conexión cliente-servidor, puertos, conexión a base de datos remota.

Habilidad para navegar en entornos web, utilizar navegadores y herramientas administrativas en línea.

4.2. Conocimientos técnicos específicos

- Para técnicos de instalación o despliegue:
    - Instalación y configuración de dependencias en Node.js y React usando npm.
    - Manejo básico de bases de datos PostgreSQL: creación de bases, ejecución de sentencias SQL, importación/exportación de datos.
    - Conocimiento del sistema de control de versiones Git, para clonar y actualizar repositorios del sistema.

- Para desarrolladores o personal de mantenimiento:
    - Lectura e interpretación de código en JavaScript (Node.js) y Dart (Flutter).
    - Conocimiento del framework Express.js para comprender la estructura del backend y sus rutas API.
    - Familiaridad con el framework Flutter para modificar la aplicación móvil y compilar en Android Studio.
    - Experiencia básica en React.js para modificar o extender el frontend web del sistema.

- Para supervisores o administradores del sistema:
    - Comprensión general de los roles de usuario, estructura de módulos y funcionamiento de cada componente (web, móvil, servidor).
    - Uso de herramientas como Postman o Insomnia para probar endpoints del sistema si es necesario.
    - Capacidad para identificar errores comunes y comunicarlos al equipo técnico.

4.3. Recomendaciones adicionales

- Tener acceso a documentación oficial de las tecnologías empleadas.
- Mantener un entorno de pruebas local para realizar cambios sin afectar el sistema en producción.
- Disponer de respaldo periódico de la base de datos y del código fuente actualizado.

5. Especificaciones técnicas

El correcto funcionamiento del sistema de gestión del Censo de Salud requiere de ciertos recursos tecnológicos, tanto a nivel de hardware como de software. Esta sección describe los requerimientos mínimos y recomendados que deben cumplir los dispositivos y entornos donde se implementarán los distintos componentes del sistema: aplicación móvil, backend (API), base de datos y plataforma web.

5.1. Requerimientos de Hardware

- Dispositivos móviles (uso en campo)

Utilizados por los encuestadores para el levantamiento de datos.

Recurso	Requerimiento mínimo	Recomendado
Sistema operativo	Android 8.0 (Oreo)	Android 11 o superior
Procesador	Quad-core 1.3 GHz	Octa-core 2.0 GHz o más
Memoria RAM	2 GB	4 GB o más
Almacenamiento libre	8 GB	16 GB o más
Conectividad	Wi-Fi o datos móviles	Wi-Fi + 4G
Pantalla	5.5 pulgadas	6.0 pulgadas

- Computador de escritorio (uso administrativo o desarrollo)

Utilizado por el personal técnico para administración web, mantenimiento y soporte.

Recurso	Requerimiento mínimo	Recomendado
Sistema operativo	Windows 10 / Ubuntu 20.04	Windows 11 / Ubuntu 22.04
Procesador	Intel Core i3	Intel Core i5 o superior
Memoria RAM	4 GB	8 GB o más
Almacenamiento	100 GB libres	256 GB SSD
Resolución de pantalla	1366 x 768	1920 x 1080
Conectividad	Red LAN o Wi-Fi estable	Conexión directa a red local

- Servidor (backend + base de datos)

Para uso en entorno de producción o pruebas centralizadas.

Recurso	Requerimiento mínimo	Recomendado
Sistema operativo	Ubuntu Server 20.04	Ubuntu Server 22.04
Procesador	2 núcleos	4 núcleos o más
Memoria RAM	4 GB	8 GB o más
Disco duro	100 GB	250 GB SSD
Seguridad	Firewall activado	Backup automático y VPN

5.2. Requerimientos de Software

Los siguientes entornos y herramientas son necesarias para la instalación, ejecución, desarrollo o mantenimiento del sistema.

Componente	Herramienta / Tecnología	Versión recomendada
Aplicación móvil	Flutter SDK	3.19.0 o superior
IDE móvil	Android Studio	Electric Eel o superior
Lenguaje móvil	Dart	Última versión estable
Backend (API REST)	Node.js	18.x LTS
Framework backend	Express.js	^4.18
Base de datos	PostgreSQL	15.x
ORM / conexión DB	pg-promise o nativa	Según implementación
Frontend web	React.js	18.x
Gestor de dependencias	npm	9.x

Editor de código	Visual Studio Code	Última versión
Control de versiones	Git	Instalado y configurado
Pruebas de carga	JMeter o K6	Opcional
Otros	JWT, dotenv, bcrypt	Para seguridad y configuración

Esta configuración técnica asegura que el sistema pueda ser instalado, desplegado y utilizado sin inconvenientes, y permite garantizar la portabilidad, estabilidad y escalabilidad del software a futuro.

6. Descripción general del sistema

El sistema de gestión del Censo de Salud del Centro de Salud Chambo es una solución informática distribuida, desarrollada con tecnologías modernas de software libre. Está conformado por tres componentes principales que interactúan entre sí bajo una arquitectura cliente-servidor:

- Aplicación móvil (Flutter)
- Servidor backend (Node.js + PostgreSQL)
- Aplicación web administrativa (React)

El sistema permite el registro, almacenamiento, consulta y análisis de datos de salud escolar, con soporte para trabajo offline y sincronización posterior. Ha sido diseñado para operar en contextos rurales, con dispositivos móviles y acceso limitado a internet, garantizando así la continuidad del proceso de censo.

6.1. Componentes del sistema

- Aplicación móvil (Flutter)
    - Utilizada por encuestadores y odontólogos en dispositivos Android.
    - Permite levantar información en zonas sin cobertura, gracias al almacenamiento local con SQLite.
    - Sincroniza automáticamente los datos al recuperar conexión a internet.
    - Cada módulo (alimentación, salud oral, visual, mental, vacunación, higiene) cuenta con su propia pantalla.
    - Control de visibilidad por tipo de rol y tipo de encuesta (salud o salud_oral).
- Backend – API REST (Node.js + Express + PostgreSQL)
    - Expone servicios RESTful para registrar, actualizar y consultar datos.
    - Implementa autenticación con JWT, control de acceso por roles y validación de datos.
    - Maneja la lógica de sincronización desde la app móvil mediante endpoints organizados por módulo.
    - Todos los datos se almacenan en una base de datos relacional PostgreSQL.

    - Incluye protección de rutas, middlewares y estructura modular (routes, controllers, models).

- Plataforma web (React)

    - Dirigida a administradores, rectores y personal de salud.
    - Permite consultar los registros por cédula, curso, unidad educativa, sexo, edad y fechas.
    - Ofrece un módulo de analítica con gráficos por módulo de salud.
    - Permite exportar información en formato Excel.
    - Estilo visual profesional, institucional y adaptable a escritorio.

6.2. Integración entre componentes

- La aplicación móvil interactúa con el servidor mediante endpoints HTTP seguros.
- Los datos capturados localmente se almacenan en SQLite y se sincronizan mediante peticiones POST al backend cuando hay conexión.
- El backend registra los datos en PostgreSQL y los pone a disposición de la plataforma web.
- La web consulta los registros a través de endpoints protegidos por JWT, renderizando los datos en tablas, gráficos y filtros dinámicos.

6.3. Arquitectura General

El sistema implementa una arquitectura de tipo cliente-servidor distribuida, con las siguientes capas:

- Capa de presentación: Flutter (cliente móvil) y React (cliente web).
- Capa de lógica de negocio: Node.js (servidor backend).
- Capa de datos: PostgreSQL (base de datos relacional).

7. Contenido del sistema

El sistema de gestión del Censo de Salud está estructurado en módulos funcionales, los cuales corresponden a áreas específicas de evaluación del estado de salud de los estudiantes. Estos módulos han sido implementados tanto en la aplicación móvil como en la plataforma web, y se encuentran organizados por encuesta, lo cual permite registrar múltiples tipos de datos para un mismo estudiante de forma independiente. Cada módulo fue diseñado siguiendo criterios técnicos del Ministerio de Salud Pública del Ecuador, y ha sido validado para su funcionamiento offline y posterior sincronización.

7.1. Módulos funcionales implementados

A continuación, se describen los principales módulos incluidos en el sistema:

- Alimentación y nutrición

    - Registro de peso y talla.
    - Cálculo automático del IMC y su clasificación.
    - Evaluación de peso para la edad y talla para la edad.
    - Uso de tablas oficiales del MSP integradas en la lógica de la app.

- Tamizaje visual

    - Evaluación de agudeza visual en ambos ojos, con y sin lentes.
    - Detección de errores de refracción según criterios médicos (20/40 o inferior).
    - Asignación automática del estado “Con error de refracción” o “Sin error”.

- Salud oral

    - Evaluación de cada pieza dental, dividida por maxilar superior e inferior.
    - Clasificación en sano, cariado, obturado, perdido, extracción indicada y no aplica.
    - Cálculo automático del total de piezas evaluadas.
    - Registro de fechas para placa bacteriana, aplicación de flúor y sellantes.

- Vacunación

    - Registro de vacunas regulares y campañas de vacunación activas.
    - Gestión de estado de aplicación: colocada, no colocada, no aplica.
    - Fechas de vacunación con validación de campañas vigentes.
    - Registro de observaciones por cada vacuna.

- Salud mental

    - Registro de factores de riesgo: violencia, consumo de sustancias, estado emocional.
    - Clasificación entre datos actuales y pasados.
    - Campos codificados con valores booleanos para análisis automático.
    - Visualización diferenciada por colores en web y app.

- Higiene y saneamiento

    - Participación del estudiante en prácticas saludables: lavado de manos, agua segura, vectores, espacios saludables.
    - Registro de número de sesiones recibidas por cada tema.
    - Evaluación simple (sí/no) para cada variable.

7.2. Otros módulos del sistema

Además de los módulos de salud, el sistema cuenta con funcionalidades adicionales para su correcta operación:

- Gestión de usuarios: creación, edición, activación/inactivación por rol.
- Gestión de unidades educativas y cursos.
- Carga masiva de estudiantes desde archivo Excel.
- Asignación de encuestas por tipo (salud, salud_oral).
- Sincronización de datos desde la app móvil.
- Visualización por estudiante con historial completo de registros.
- Módulo de analítica por unidad educativa, curso, sexo, edad y fecha.

8. Funciones principales del sistema

El sistema informático desarrollado para la gestión del Censo de Salud cumple con una serie de funciones clave que permiten cubrir todo el ciclo de levantamiento, almacenamiento, consulta y análisis de la información de salud escolar.

Estas funciones están distribuidas entre los tres componentes del sistema: aplicación móvil, servidor backend y plataforma web, los cuales trabajan de forma integrada bajo un esquema cliente-servidor.

8.1. Funciones generales del sistema

- Registro de encuestas de salud por estudiante, clasificadas por tipo (salud o salud_oral).
- Gestión de módulos clínicos independientes para cada área de evaluación (nutrición, salud mental, etc.).
- Soporte para trabajo en modo offline mediante almacenamiento local en SQLite (móvil).
- Sincronización automática de datos al recuperar conexión.
- Autenticación segura mediante tokens JWT.
- Control de acceso por roles, diferenciando permisos para encuestadores, odontólogos, rectores y administradores.
- Generación de informes y gráficos estadísticos por módulo y unidad educativa.
- Exportación de información en formato Excel desde la web.
- Carga masiva de estudiantes y unidades educativas desde archivos .xlsx.

8.2. Funciones específicas por componente

- **Aplicación móvil (Flutter)**
    - Acceso mediante login protegido con JWT.
    - Visualización de datos del estudiante antes de registrar.
    - Formulario de ingreso por módulo con validaciones automáticas.
    - Almacenamiento local por módulo usando SQLite.
    - Sincronización modular de registros al backend.
    - Interfaz responsiva y adaptada al contexto rural.

- Control de visibilidad según rol y tipo de encuesta.

- **Backend (Node.js + PostgreSQL)**

  - Exposición de endpoints RESTful para cada módulo.
  - Verificación de identidad y roles mediante middleware.
  - Validación de datos antes de ser almacenados.
  - Relación entre estudiantes, encuestas y módulos a través de claves foráneas.
  - Lógica de sincronización que evita duplicados.
  - Registro de campañas de vacunación y asignación por edad.
  - Estructura modular por carpeta (routes, controllers, models).

- **Plataforma Web (React)**

  - Login con verificación de sesión y roles.
  - Consulta de registros de salud por cédula, unidad o curso.
  - Módulo de analítica con filtros por sexo, edad, curso, fechas y unidad educativa.
  - Gráficos por módulo: IMC, salud oral, visual, mental, vacunación, higiene.
  - Exportación de datos fuera de rango a Excel.
  - Gestión visual de usuarios (activar, editar, reasignar).
  - Diseño moderno, accesible y profesional.

9. Arquitectura del sistema

El sistema de gestión del Censo de Salud está construido bajo una arquitectura cliente-servidor distribuida, compuesta por tres capas funcionales que interactúan entre sí mediante servicios REST. Esta estructura modular facilita el mantenimiento, escalabilidad y reutilización del sistema, permitiendo su uso tanto en entornos locales como remotos.

9.1. Componentes principales de la arquitectura

- **Aplicación Móvil (Cliente 1)**

  - Desarrollada en Flutter con lenguaje Dart.
  - Funciona en modo offline, utilizando SQLite para almacenamiento local.
  - Realiza peticiones HTTP al backend para sincronización.
  - Diseñada para dispositivos Android, con compatibilidad desde Android 8.0.

- **Servidor Backend (API REST)**

  - Desarrollado en Node.js utilizando el framework Express.js.
  - Provee servicios RESTful protegidos con JWT.
  - Realiza validaciones, operaciones CRUD, y maneja lógica de negocio.
  - Administra el flujo de sincronización desde la app móvil.
  - Se conecta a la base de datos PostgreSQL para almacenamiento persistente.

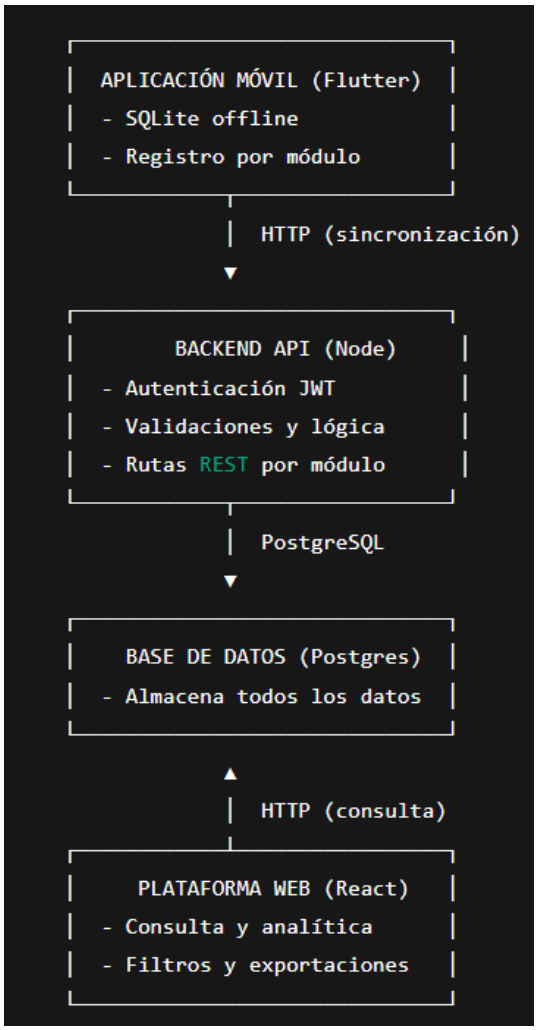
- **Plataforma Web (Cliente 2)**

    - Desarrollada en React.js.
    - Se conecta a los endpoints del backend mediante peticiones HTTP autenticadas.
    - Permite la visualización de registros, exportación de datos y acceso a gráficos analíticos.
    - Aplicación SPA (Single Page Application) con gestión de rutas y sesiones por rol.

- **Base de Datos (Servidor de datos)**

    - Motor: PostgreSQL
    - Base relacional compuesta por más de 15 tablas (usuarios, estudiantes, encuestas, módulos).
    - Utiliza claves primarias UUID y relaciones con claves foráneas.
    - Todas las operaciones de lectura y escritura son controladas por el backend.

9.2. Diagrama lógico de arquitectura



9.3. Flujo de sincronización

- El encuestador registra datos en la app (sin conexión) → guardado en SQLite.
- Cuando el dispositivo detecta conexión a internet, envía los datos al backend mediante una petición POST.
- El backend valida, guarda en PostgreSQL y responde con éxito.
- La app marca ese registro como sincronizado.
- La plataforma web accede al backend y lee la información sincronizada para su visualización y análisis.

Esta arquitectura permite que el sistema sea robusto, modular y escalable, soportando múltiples usuarios en campo y consultas simultáneas desde la plataforma web.

10. Base de datos

El sistema de Censo de Salud utiliza una base de datos relacional construida sobre PostgreSQL, la cual actúa como repositorio central de toda la información capturada desde la aplicación móvil y consultada desde la plataforma web.

La base está estructurada bajo el esquema public y contiene más de 15 tablas principales, que representan usuarios, unidades educativas, estudiantes, encuestas y todos los módulos clínicos implementados (alimentación, tamizaje visual, salud oral, etc.).

Cada tabla fue diseñada con tipos de datos adecuados, identificadores únicos (UUID), claves foráneas para mantener integridad referencial y convenciones normalizadas para escalabilidad y mantenibilidad.

10.1. Diseño general

- Modelo relacional: basado en entidades y relaciones normalizadas.
- Identificadores únicos: uso de uuid como claves primarias para garantizar unicidad entre dispositivos móviles y servidor.
- Relaciones: uso de claves foráneas (FOREIGN KEY) entre estudiantes, encuestas y módulos de salud.
- Integridad: aplicación de restricciones NOT NULL, UNIQUE y tipos específicos como boolean, timestamp, numeric, varchar.

10.2. Estructura de tablas principales

A continuación, se presenta la descripción técnica de las tablas más importantes del sistema:

10.2.1. Tabla: usuarios

Contiene la información de los usuarios registrados en el sistema, como encuestadores, odontólogos, administradores y rectores.

Campo	Tipo de Dato	Descripción
id	uuid	Identificador único del usuario
cedula	varchar(10)	Cédula de identidad
nombres	varchar(100)	Nombres del usuario
apellidos	varchar(100)	Apellidos del usuario
correo	varchar	Correo institucional
password	text	Contraseña encriptada
rol_id	uuid	Relación con la tabla roles
carrera	varchar(100)	Carrera profesional
fecha_nacimiento	date	Fecha de nacimiento
genero	varchar(20)	Género (masculino/femenino/otro)
area_trabajo	varchar(100)	Área de trabajo
estado	varchar	Estado del usuario (Activo/Inactivo)
fecha_creacion	timestamp	Fecha de registro

10.2.2. Tabla: roles

Define los tipos de usuario existentes en el sistema.

Campo	Tipo de Dato	Descripción
id	uuid	Identificador único del rol
nombre	varchar(50)	Nombre del rol (admin, encuestador, etc.)

0.2.3. Tabla: unidades_educativas

Contiene el listado de unidades educativas participantes en el censo.

Campo	Tipo de Dato	Descripción
id	uuid	ID de la unidad educativa
nombre	varchar(150)	Nombre completo de la institución
direccion	text	Dirección física
tipo	varchar(50)	Tipo (Fiscal, Particular, etc.)
fecha_creacion	timestamp	Fecha de registro

10.2.4. Tabla: estudiantes

Contiene la información detallada de los estudiantes registrados manualmente en el sistema.

Campo	Tipo de Dato	Descripción
cedula	varchar(10)	Cédula del estudiante
unidad_id	uuid	Relación con la unidad educativa (FK)
nombres	varchar(100)	Nombres del estudiante
apellidos	varchar(100)	Apellidos del estudiante
fecha_nacimiento	date	Fecha de nacimiento
sexo	varchar(10)	Género
discapacidad	varchar(50)	Tipo de discapacidad, si aplica
alergias	text	Alergias reportadas
tipo_alergia	text	Detalle específico de la alergia
representante_nombre	varchar(100)	Nombre del representante legal
representante_telefono	varchar(20)	Teléfono del representante
parentesco_representante	text	Parentesco del representante
comunidad_residencia	varchar(100)	Comunidad o barrio donde reside el estudiante
fecha_registro	timestamp	Fecha de registro del estudiante
profesor_nombre	varchar(100)	Nombre del docente responsable
profesor_cedula	varchar(20)	Cédula del docente

posee_seguro	text	Indica si posee seguro (Sí/No)
tipo_seguro	text	Tipo de seguro (IESS, MSP, privado, etc.)
grado_paralelo	text	Curso y paralelo del estudiante

10.2.5. Tabla: estudiantes_precargados

Tabla auxiliar utilizada para carga masiva de estudiantes desde archivos Excel.

Campo	Tipo de Dato	Descripción
cedula	varchar(20)	Cédula del estudiante
nombres	varchar(100)	Nombres
apellidos	varchar(100)	Apellidos
fecha_nacimiento	date	Fecha de nacimiento
sexo	varchar(10)	Género
discapacidad	varchar(50)	Tipo de discapacidad
alergias	text	Alergias
tipo_alergia	text	Tipo de alergia
nombre_representante	varchar(100)	Nombre del representante legal
telefono_representante	varchar(20)	Teléfono del representante
parentesco_representante	text	Parentesco del representante
comunidad_residencia	varchar(100)	Comunidad
profesor_nombre	varchar(100)	Docente a cargo
profesor_cedula	varchar(20)	Cédula del docente
posee_seguro	text	Posee seguro (Sí/No)
tipo_seguro	text	Tipo de seguro
grado_paralelo	text	Grado y paralelo del estudiante

10.2.6. Tabla: encuestas

Registra cada levantamiento de información realizado a un estudiante. Relaciona un estudiante con los módulos de salud.

Campo	Tipo de Dato	Descripción
id	uuid	Identificador único de la encuesta
usuario_id	uuid	Usuario que realizó el registro (FK a usuarios)
cedula_estudiante	varchar(10)	Cédula del estudiante evaluado
fecha	timestamp	Fecha de la encuesta
tipo_encuesta	varchar(50)	salud o salud_oral

10.2.7. Tabla: alimentacion_nutricion

Almacena los datos nutricionales obtenidos del estudiante, incluyendo peso, talla, cálculo del IMC y clasificaciones según edad y sexo.

Campo	Tipo de Dato	Descripción
id	uuid	Identificador único del registro
encuesta_id	uuid	Relación con la encuesta correspondiente
peso	numeric(5,2)	Peso en kilogramos
talla	numeric(5,2)	Talla en metros
imc	numeric(5,2)	Índice de Masa Corporal
estado_imc	varchar(50)	Clasificación según IMC (bajo, normal, etc.)
peso_para_edad	varchar(50)	Clasificación de peso para edad
talla_para_edad	varchar(50)	Clasificación de talla para edad

10.2.8. Tabla: tamizaje_visual

Registra la evaluación de agudeza visual de cada ojo con y sin lentes, junto con el resultado del tamizaje.

Campo	Tipo de Dato	Descripción
id	uuid	Identificador único del registro
encuesta_id	uuid	Relación con encuesta (FK)
ojo_izquierdo_sin_lentes	varchar(30)	Agudeza ojo izquierdo sin lentes
ojo_derecho_sin_lentes	varchar(30)	Agudeza ojo derecho sin lentes
ojo_izquierdo_con_lentes	varchar(30)	Agudeza ojo izquierdo con lentes
ojo_derecho_con_lentes	varchar(30)	Agudeza ojo derecho con lentes
resultado_sin_lentes	text	Resultado general sin lentes
resultado_con_lentes	text	Resultado general con lentes

10.2.9. Tabla: salud_oral

Guarda la información dental del estudiante, con separación por maxilar, tipo de afectación y fechas de prevención bucal.

Campo	Tipo de Dato	Descripción
id	uuid	ID del registro
encuesta_id	uuid	Relación con encuesta (FK)
placa_bacteriana	boolean	¿Tiene placa bacteriana?
aplicacion_fluor	boolean	¿Se aplicó flúor?
aplicacion_sellantes	boolean	¿Se aplicaron sellantes?
fecha_placa_bacteriana	date	Fecha de control de placa bacteriana
fecha_aplicacion_fluor	date	Fecha de aplicación de flúor
fecha_aplicacion_sellantes	date	Fecha de aplicación de sellantes
observaciones	text	Observaciones del odontólogo
piezas_sanas_superior	integer	Número de piezas sanas (maxilar superior)
piezas_sanas_inferior	integer	Piezas sanas (maxilar inferior)
piezas_cariadas_superior	integer	Cariadas en el maxilar superior
piezas_cariadas_inferior	integer	Cariadas en el maxilar inferior
piezas_obturadas_superior	integer	Obturadas (superior)
piezas_obturadas_inferior	integer	Obturadas (inferior)
piezas_perdidas_superior	integer	Perdidas (superior)
piezas_perdidas_inferior	integer	Perdidas (inferior)
extraccion_indicada_superior	text	Piezas indicadas para extracción (sup.)
extraccion_indicada_inferior	text	Piezas indicadas para extracción (inf.)

10.2.10. Tabla: vacunacion

Registra las vacunas aplicadas a cada estudiante, ya sea de forma regular o por campaña.

Campo	Tipo de Dato	Descripción
id	uuid	Identificador único del registro
encuesta_id	uuid	Relación con la encuesta correspondiente (FK)
vacuna	varchar(50)	Nombre de la vacuna aplicada
fecha_vacunacion	date	Fecha en que se colocó la vacuna
tipo	varchar(50)	Tipo de vacuna: Regular o Campaña
estado	varchar(50)	Estado de aplicación: Colocada, No colocada, etc.

observacion	text	Observaciones adicionales

10.2.11. Tabla: campanias_vacunacion

Define las campañas de vacunación habilitadas, su rango de edad y la vacuna asociada.

Campo	Tipo de Dato	Descripción
id	uuid	ID de la campaña
nombre	varchar(100)	Nombre de la campaña
vacuna	varchar(100)	Vacuna administrada en la campaña
edad_desde_anios	integer	Edad mínima en años para aplicar la campaña
edad_hasta_anios	integer	Edad máxima en años
created_at	timestamp	Fecha de creación de la campaña

10.2.12. Tabla: registro_campania

Relación entre una encuesta y una campaña de vacunación aplicada.

Campo	Tipo de Dato	Descripción
id	uuid	ID del registro
encuesta_id	uuid	Relación con encuesta (FK)
campania_id	uuid	Relación con campaña (FK)
fecha_vacunacion	text	Fecha de vacunación en campaña
created_at	timestamp	Fecha de creación del registro

10.2.13. Tabla: salud_mental

Almacena indicadores sobre salud emocional y conductas de riesgo del estudiante, categorizadas por momento (pasado/actual) y tipo (violencia, consumo, emocional).

Campo	Tipo de Dato	Descripción
id	uuid	ID del registro
encuesta_id	uuid	Relación con la encuesta (FK)
violencia_fisica_familiar	boolean	Ha sufrido violencia física en la familia
violencia_fisica_escolar	boolean	Violencia física en la escuela
violencia_fisica_otro	boolean	Violencia física en otro ámbito
violencia_psicologica_familiar	boolean	Violencia psicológica en la familia
violencia_psicologica_escolar	boolean	Violencia psicológica en la escuela
violencia_psicologica_otro	boolean	Violencia psicológica en otro ámbito
violencia_sexual_familiar	boolean	Violencia sexual en la familia
violencia_sexual_escolar	boolean	Violencia sexual en la escuela
violencia_sexual_otro	boolean	Violencia sexual en otro contexto
alcohol_pasado	boolean	Consumo de alcohol en el pasado
alcohol_actual	boolean	Consumo de alcohol actual
tabaco_pasado	boolean	Consumo de tabaco en el pasado
tabaco_actual	boolean	Consumo actual de tabaco
drogas_pasado	boolean	Consumo de drogas en el pasado
drogas_actual	boolean	Consumo de drogas actual
ansiedad_pasada	boolean	Presencia de ansiedad en el pasado
ansiedad_actual	boolean	Ansiedad actual
depresion_pasada	boolean	Depresión pasada
depresion_actual	boolean	Depresión actual
intento_suicidio_pasado	boolean	Intento de suicidio en el pasado

intento_suicidio_actual	boolean	Intento de suicidio actual

10.2.14. Tabla: higiene_saneamiento

Evalúa la participación del estudiante en prácticas de higiene y ambientes saludables.

Campo	Tipo de Dato	Descripción
id	uuid	ID del registro
encuesta_id	uuid	Relación con encuesta (FK)
participa_riesgos_vectores	boolean	Participa en actividades contra vectores
participa_espacios_saludables	boolean	Participa en ambientes saludables
participa_lavado_manos	boolean	Participa en lavado de manos
participa_agua_segura	boolean	Participa en consumo de agua segura
numero_vectores	integer	Número de sesiones sobre vectores
numero_espacios_saludables	integer	Sesiones sobre espacios saludables
numero_lavado_manos	integer	Sesiones de lavado de manos
numero_agua_segura	integer	Sesiones sobre agua segura

11. Backend del sistema (NODE.JS)

El backend del sistema fue desarrollado utilizando Node.js con el framework Express.js, bajo una arquitectura modular y orientada a servicios REST. Este componente es responsable de procesar las peticiones de la aplicación móvil y de la plataforma web, aplicando la lógica de negocio, validaciones, seguridad, y conectividad con la base de datos PostgreSQL.

Está diseñado para ser escalable, seguro y fácilmente mantenible, utilizando estándares modernos como autenticación con JWT, control de roles, y una estructura clara por carpetas y controladores.

11.1. Estructura general del proyecto

El backend está organizado en carpetas según la responsabilidad de cada parte del

sistema:
backend/
├── controllers/ ← Lógica de negocio de cada módulo
├── routes/ ← Definición de rutas API
├── models/ ← Conexión con la base de datos
├── middlewares/ ← Autenticación y control de acceso
├── config/ ← Conexiones y configuración global
├── utils/ ← Funciones auxiliares
├── services/ ← Reutilización de lógica específica
├── app.js ← Configuración principal de Express
├── server.js ← Punto de entrada del backend
└── .env ← Variables de entorno (.env)

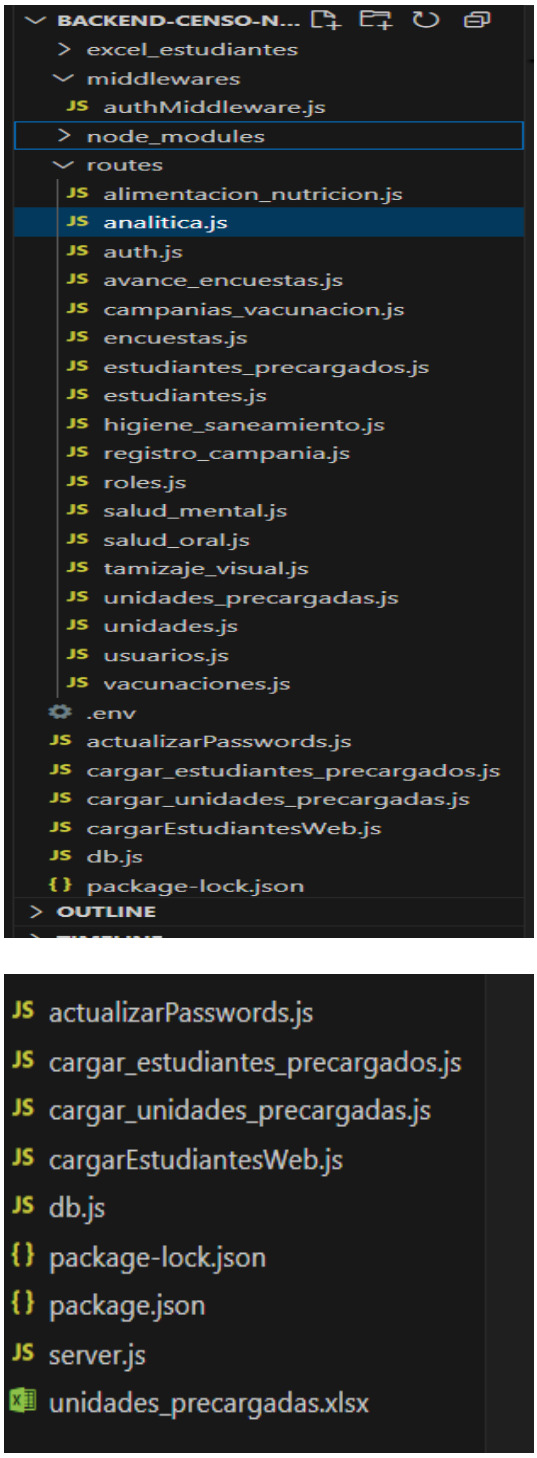


Figura 1. Estructura de carpetas del backend

11.2. Autenticación y seguridad

- El backend utiliza tokens JWT (JSON Web Token) para autenticar a los usuarios.
- Al iniciar sesión, el usuario recibe un token que debe enviar en cada petición protegida.
- El middleware verificarToken se encarga de validar la autenticidad y vigencia del token.

- El middleware verificarRol([‘admin’, ‘encuestador’]) restringe el acceso a rutas según el rol.

📌 Estos middlewares están definidos en la carpeta /middlewares/.

11.3. Rutas principales (endpoints REST)

Cada módulo funcional del sistema (nutrición, salud oral, etc.) tiene un archivo de rutas ubicado en la carpeta routes/. Estas rutas son consumidas por la app móvil y por la web.

Ejemplo (módulo alimentación):

```
const express = require("express");

const router = express.Router();

const pool = require("../db"); // Conexión a PostgreSQL

const { verificarToken, verificarRol } = require("../middlewares/authMiddleware");

// Función para validar UUIDs

const esUUIDValido = (id) =>

                    /^[0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}$/.test(id);

// ☑ Obtener todos los registros

router.get("/", verificarToken, verificarRol(["admin", "encuestador"]), async (req, res) => {

  try {

    const result = await pool.query("SELECT * FROM alimentacion_nutricion ORDER BY id ASC");

    res.json(result.rows);

  } catch (err) {

    console.error("❌ Error en GET /alimentacion_nutricion:", err.message);

    res.status(500).json({ error: "Error en el servidor" });

  }
```

```
});

// ☑ Obtener un registro por ID

router.get("/:id", verificarToken, verificarRol(["admin", "encuestador"]), async (req, res) =>
{

  try {

    const { id } = req.params;

    if (!esUUIDValido(id)) {

      return res.status(400).json({ error: "ID no válido" });

    }

    const result = await pool.query("SELECT * FROM alimentacion_nutricion WHERE id =
$1", [id]);



    if (result.rows.length === 0) {

      return res.status(404).json({ error: "Registro no encontrado" });

    }

    res.json(result.rows[0]);

  } catch (err) {

    console.error("❌ Error en GET /alimentacion_nutricion/:id:", err.message);

    res.status(500).json({ error: "Error en el servidor" });

  }

});

// ☑ Obtener TODOS los registros por cédula

router.get("/cedula/:cedula", verificarToken, verificarRol(["admin", "rector"]), async (req,
res) => {

  try {

    const { cedula } = req.params;
```

```
    const result = await pool.query(`
      SELECT an.*, e.nombres, e.apellidos
      FROM alimentacion_nutricion an
      JOIN encuestas enc ON enc.id = an.encuesta_id
      JOIN estudiantes e ON e.cedula = enc.cedula_estudiante
      WHERE e.cedula = $1
      ORDER BY an.id DESC
    `, [cedula]);
    if (result.rows.length === 0) {
        return res.status(404).json({ mensaje: "No hay registros de alimentación para esta cédula" });
    }
    res.json(result.rows); // ☑ Devolver todos los registros
  } catch (err) {
    console.error("❌ Error en GET /alimentacion_nutricion/cedula/:cedula:", err.message);
    res.status(500).json({ error: "Error en el servidor" });
  }
});
// ☑ Crear nuevo registro
router.post("/", verificarToken, verificarRol(["admin", "encuestador"]), async (req, res) => {
  try {
    const {
      encuesta_id, peso, talla, imc,
      estado_imc, peso_para_edad, talla_para_edad
    } = req.body;
```

```
    if (!esUUIDValido(encuesta_id)) {
      return res.status(400).json({ error: "El ID de la encuesta no es válido" });
    }
   const encuestaExiste = await pool.query("SELECT id FROM encuestas WHERE id = $1",
[encuesta_id]);
    if (encuestaExiste.rows.length === 0) {
      return res.status(400).json({ error: "La encuesta no existe" });
    }
    const result = await pool.query(
      `INSERT INTO alimentacion_nutricion (
        id, encuesta_id, peso, talla, imc,
        estado_imc, peso_para_edad, talla_para_edad
      ) VALUES (
        gen_random_uuid(), $1, $2, $3, $4, $5, $6, $7
      ) RETURNING *`,
      [encuesta_id, peso, talla, imc, estado_imc, peso_para_edad, talla_para_edad]
    );
    res.status(201).json(result.rows[0]);
  } catch (err) {
    console.error("❌ Error en POST /alimentacion_nutricion:", err.message);
    res.status(500).json({ error: "Error en el servidor" });
  }
});
// ☑ Actualizar un registro
router.put("/:id", verificarToken, verificarRol(["admin"]), async (req, res) => {
  try {
```

```
    const { id } = req.params;
    const {
      peso, talla, imc,
      estado_imc, peso_para_edad, talla_para_edad
    } = req.body;
    if (!esUUIDValido(id)) {
      return res.status(400).json({ error: "ID no válido" });
    }
    const existe = await pool.query("SELECT id FROM alimentacion_nutricion WHERE id
= $1", [id]);
    if (existe.rows.length === 0) {
      return res.status(404).json({ error: "Registro no encontrado" });
    }
    const result = await pool.query(
      `UPDATE alimentacion_nutricion SET
        peso = $1,
        talla = $2,
        imc = $3,
        estado_imc = $4,
        peso_para_edad = $5,
        talla_para_edad = $6
       WHERE id = $7 RETURNING *`,
      [peso, talla, imc, estado_imc, peso_para_edad, talla_para_edad, id]
    );
    res.json(result.rows[0]);
  } catch (err) {
```

```
    console.error("❌ Error en PUT /alimentacion_nutricion/:id:", err.message);
    res.status(500).json({ error: "Error en el servidor" });
  }
});
// ☑ Eliminar un registro
router.delete("/:id", verificarToken, verificarRol(["admin"]), async (req, res) => {
  try {
    const { id } = req.params;
    if (!esUUIDValido(id)) {
      return res.status(400).json({ error: "ID no válido" });
    }
    const existe = await pool.query("SELECT id FROM alimentacion_nutricion WHERE id = $1", [id]);
    if (existe.rows.length === 0) {
      return res.status(404).json({ error: "Registro no encontrado" });
    }
    await pool.query("DELETE FROM alimentacion_nutricion WHERE id = $1", [id]);
    res.json({ message: "Registro eliminado correctamente" });
  } catch (err) {
    console.error("❌ Error en DELETE /alimentacion_nutricion/:id:", err.message);
    res.status(500).json({ error: "Error en el servidor" });
  }
});
// ☑ Sincronizar registros desde Flutter
router.post("/sincronizar", verificarToken, verificarRol(["admin", "encuestador", "odontologo"]), async (req, res) => {
```

```
 try {
   const { registros } = req.body;
   if (!Array.isArray(registros) || registros.length === 0) {
     return res.status(400).json({ error: "No se enviaron registros válidos" });
   }
   let insertados = 0;
   for (const registro of registros) {
     const {
       id, encuesta_id, peso, talla, imc,
       estado_imc, peso_para_edad, talla_para_edad
     } = registro;
     if (!esUUIDValido(id) || !esUUIDValido(encuesta_id)) {
       console.warn(`⛔ ID inválido: ${id} o ${encuesta_id}`);
       continue;
     }
      const encuestaExiste = await pool.query("SELECT id FROM encuestas WHERE id =
$1", [encuesta_id]);
     if (encuestaExiste.rows.length === 0) {
       console.warn(`❌ La encuesta ${encuesta_id} no existe (registro omitido)`);
       continue;
     }
    const yaExiste = await pool.query("SELECT id FROM alimentacion_nutricion WHERE
id = $1", [id]);
     if (yaExiste.rows.length > 0) {
       console.log(`⚠ Registro duplicado (omitido): ${id}`);
       continue;
```

```
    }

    await pool.query(

      `INSERT INTO alimentacion_nutricion (

        id, encuesta_id, peso, talla, imc,

        estado_imc, peso_para_edad, talla_para_edad

      ) VALUES ($1, $2, $3, $4, $5, $6, $7, $8)`,

      [id, encuesta_id, peso, talla, imc, estado_imc, peso_para_edad, talla_para_edad]

    );

    insertados++;

  }

    res.json({ mensaje: `☑ ${insertados} registros de alimentación sincronizados correctamente` });

 } catch (err) {

  console.error("❌ Error en POST /alimentacion_nutricion/sincronizar:", err.message);

  res.status(500).json({ error: "Error al sincronizar registros de alimentación" });

 }

});

module.exports = router;
```

11.4. Lógica de sincronización

Cada módulo incluye una ruta tipo POST /modulo/sincronizar, utilizada por la app móvil para enviar los datos almacenados localmente cuando se recupera la conexión a internet.

Proceso:

- La app envía los datos a POST /modulo/sincronizar.
- El backend verifica si ya existe un registro para esa encuesta_id.
- Si no existe, **crea** el registro.
- Si existe, **actualiza** el registro existente.
- Retorna confirmación para marcar como sincronizado en la app.

Este mecanismo evita duplicación de datos y permite sincronización módulo por módulo.

11.5. Conexión a la base de datos (PostgreSQL)

- El backend se conecta a PostgreSQL usando el módulo pg o pg-promise.
- La configuración de conexión está definida en el archivo /config/db.js.
- Las operaciones SQL se realizan directamente o a través de funciones en /models/.

Ejemplo de conexión:

```
const { Pool } = require("pg");

require("dotenv").config();

const pool = new Pool({

   user: process.env.DB_USER,

   host: process.env.DB_HOST,

   database: process.env.DB_NAME,

   password: process.env.DB_PASSWORD,

   port: process.env.DB_PORT

});


pool.connect()

   .then(() => console.log("☑ Conexión exitosa a PostgreSQL"))

   .catch(err => console.error("❌ Error de conexión a PostgreSQL", err));

module.exports = pool;
```

11.6. Ejecución del backend

El backend se ejecuta desde la raíz del proyecto con el siguiente comando:

- node server.js

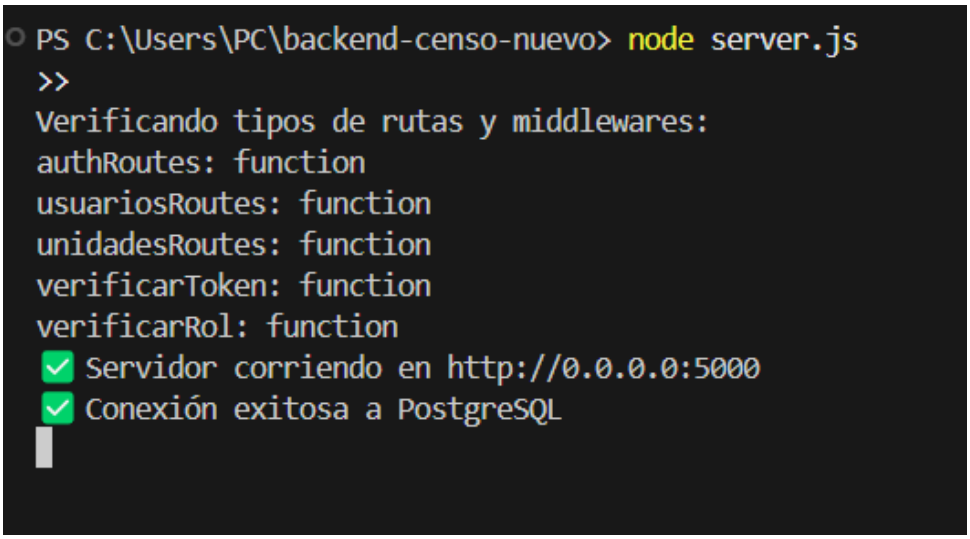


Figura 2. Backend ejecutándose correctamente en consola

12. Aplicación móvil (Flutter)

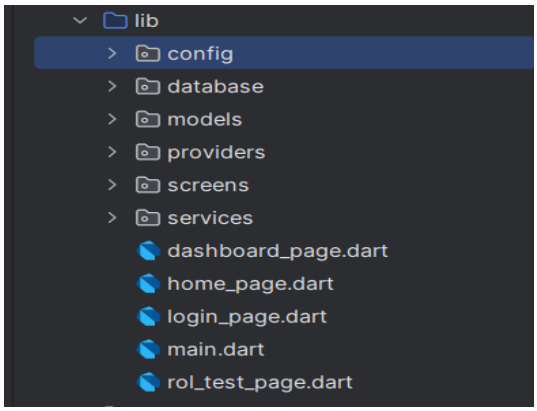
La aplicación móvil del sistema de Censo de Salud fue desarrollada en Flutter, un framework multiplataforma de código abierto que permite compilar aplicaciones nativas para Android desde una única base de código en Dart.

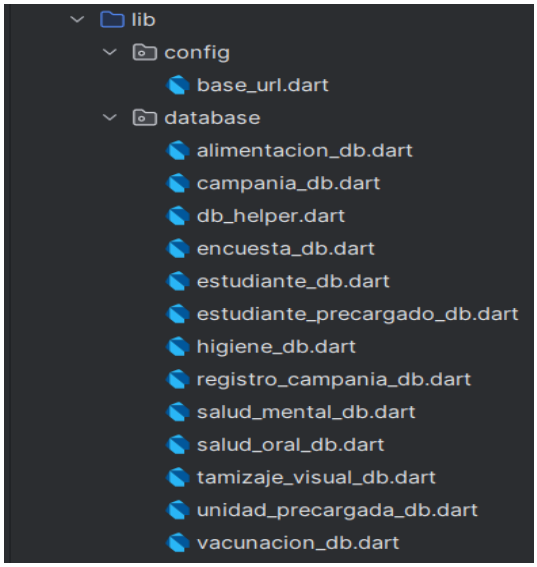
Esta app está diseñada para ser utilizada por personal de salud en campo (encuestadores u odontólogos), y se caracteriza por su capacidad de funcionar completamente sin conexión a internet, almacenando los registros localmente mediante SQLite y sincronizándolos luego con el servidor backend cuando se restablece la conexión.

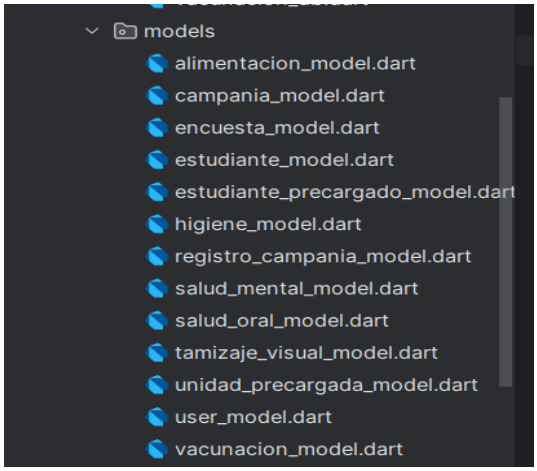
12.1. Estructura del proyecto

El proyecto Flutter se organiza en carpetas según funciones:

lib/	
├── main.dart	
├── screens/	← Pantallas de cada módulo
├── models/	← Modelos de datos por módulo
├── database/	← Funciones SQLite locales
├── services/	← Lógica de sincronización y API
├── widgets/	← Componentes visuales reutilizables
├── utils/	← Funciones de ayuda (edad, colores, etc.)
├── login/	← Pantalla y lógica de autenticación
└── dashboard/	← Menú principal y navegación


lib
config
database
models
providers
screens
services
dashboard_page.dart
home_page.dart
login_page.dart
main.dart
rol_test_page.dart


lib
config
base_url.dart
database
alimentacion_db.dart
campania_db.dart
db_helper.dart
encuesta_db.dart
estudiante_db.dart
estudiante_precargado_db.dart
higiene_db.dart
registro_campania_db.dart
salud_mental_db.dart
salud_oral_db.dart
tamizaje_visual_db.dart
unidad_precargada_db.dart
vacunacion_db.dart


models
alimentacion_model.dart
campania_model.dart
encuesta_model.dart
estudiante_model.dart
estudiante_precargado_model.dart
higiene_model.dart
registro_campania_model.dart
salud_mental_model.dart
salud_oral_model.dart
tamizaje_visual_model.dart
unidad_precargada_model.dart
user_model.dart
vacunacion_model.dart

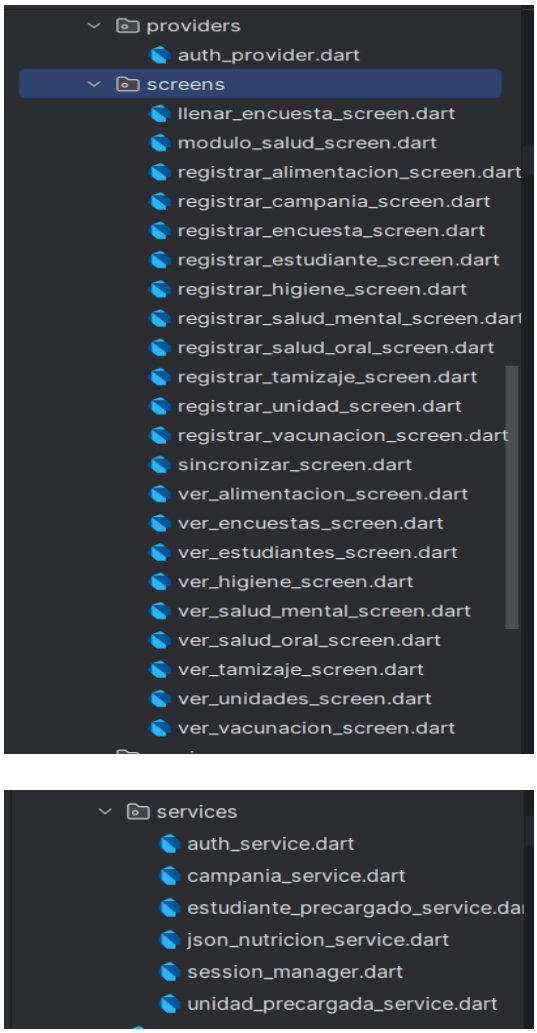


Figura 4. Estructura de carpetas del proyecto flutter

12.2. Registro offline y almacenamiento local

Una de las funcionalidades clave de la aplicación móvil es su capacidad para operar sin conexión a internet, lo cual es esencial en entornos rurales o lugares donde la conectividad es limitada o intermitente. Para lograr esto, se implementó una estrategia de persistencia local de datos utilizando SQLite, una base de datos ligera y embebida soportada por Flutter a través del paquete sqflite.

12.2.1. ¿Cómo funciona el almacenamiento local?

- Cada módulo de salud (alimentación, salud oral, etc.) tiene una tabla local SQLite independiente.
- Cuando un usuario completa un formulario y pulsa “Guardar”, el registro no se envía inmediatamente al servidor, sino que se inserta en la base local SQLite.
- Esta inserción se realiza usando funciones insert() desde la clase DatabaseHelper o equivalentes, por ejemplo:

```
await db.insert('tamizaje_visual', {
```

```
  'encuesta_id': idEncuesta,
  'ojo_izquierdo_sin_lentes': valor1,
  'ojo_derecho_sin_lentes': valor2,
  'resultado_sin_lentes': resultadoFinal,
  'sincronizado': 0 // 0 = pendiente de sincronizar
});
```

Todos los registros insertados localmente tienen un campo sincronizado que indica si ese dato ya fue enviado al backend (0 = no, 1 = sí).

12.2.2. Estructura de la base local

Cada tabla en SQLite replica parcialmente la estructura de su equivalente en PostgreSQL, manteniendo los campos necesarios para:

- Registrar los datos capturados en el formulario
- Relacionar con la encuesta principal mediante encuesta_id
- Marcar el estado de sincronización
- En algunos casos, llevar un timestamp local de creación

Ejemplo de tabla local salud_mental:

```
import 'package:sqflite/sqflite.dart';
import 'package:censo_salud/database/db_helper.dart';
import 'package:censo_salud/models/salud_mental_model.dart';
class SaludMentalDB {
  static const String tabla = 'salud_mental';
  //  ◇ Crear la tabla (estructura según la base de datos PostgreSQL)
  static Future<void> crearTabla(Database db) async {
    await db.execute('''
      CREATE TABLE IF NOT EXISTS $tabla (
        id TEXT PRIMARY KEY,
        encuesta_id TEXT,
```

```
        violencia_fisica_familiar INTEGER,
        violencia_fisica_escolar INTEGER,
        violencia_fisica_otro INTEGER,
        violencia_psicologica_familiar INTEGER,
        violencia_psicologica_escolar INTEGER,
        violencia_psicologica_otro INTEGER,
        violencia_sexual_familiar INTEGER,
        violencia_sexual_escolar INTEGER,
        violencia_sexual_otro INTEGER,
        alcohol_pasado INTEGER,
        alcohol_actual INTEGER,
        tabaco_pasado INTEGER,
        tabaco_actual INTEGER,
        drogas_pasado INTEGER,
        drogas_actual INTEGER,
        ansiedad_pasada INTEGER,
        ansiedad_actual INTEGER,
        depresion_pasada INTEGER,
        depresion_actual INTEGER,
        intento_suicidio_pasado INTEGER,
        intento_suicidio_actual INTEGER
    )
  ''');
}
// ◇ Insertar nuevo registro
```

```
  static Future<void> insertar(SaludMentalModel model) async {
    final db = await DBHelper.database;
    await db.insert(
      tabla,
      model.toMap(),
      conflictAlgorithm: ConflictAlgorithm.replace,
    );
    print("☑ Registro de salud mental insertado localmente: ${model.id}");
  }

  // ◇ Obtener todos los registros
  static Future<List<SaludMentalModel>> obtenerTodos() async {
    final db = await DBHelper.database;
    final res = await db.query(tabla);
    return res.map((e) => SaludMentalModel.fromMap(e)).toList();
  }

  // ◇ Eliminar todos los registros (después de sincronizar)
  static Future<void> eliminarTodos() async {
    final db = await DBHelper.database;
    await db.delete(tabla);
    print("🗑 Registros de salud mental eliminados localmente");
  }
}
```

12.2.3. Ventajas de esta implementación

- Independencia de conectividad: la app puede funcionar en cualquier lugar.
- Persistencia confiable: incluso si la app se cierra, los datos se mantienen.

- Control de errores: si ocurre un error en la sincronización, el registro permanece en SQLite y puede reenviarse más tarde.
- Facilidad de sincronización modular: cada módulo se puede sincronizar por separado, evitando colapsos por lotes masivos.

12.2.4. Gestión desde Flutter

- Se implementa un servicio por módulo (AlimentacionService, VisualService, etc.) que contiene las funciones para:
    - Guardar localmente (guardarLocal)
    - Leer registros pendientes (obtenerNoSincronizados)
    - Marcar como sincronizado (actualizarEstado)
- Estas funciones acceden a la base local mediante la instancia global del helper (DatabaseHelper.instance).

En resumen:

Cada vez que el encuestador completa un módulo, los datos se guardan localmente en SQLite con sincronizado = 0. Luego, un proceso posterior se encarga de sincronizarlos al backend cuando haya conexión disponible.

12.3. Sincronización con el backend

La sincronización de datos entre la aplicación móvil y el servidor se realiza mediante el envío de registros locales almacenados en SQLite hacia la API RESTful desarrollada en Node.js. Este proceso está diseñado para ser modular, controlado y tolerante a errores, permitiendo que los registros se transmitan uno por uno y por módulo.

12.3.1. ¿Cuándo se sincroniza?

La sincronización se puede activar:

- De forma automática, al acceder a la pantalla de sincronización (SincronizarScreen) cuando hay conexión.
- O manualmente, en caso de que se implemente un botón de "Sincronizar ahora" (opcional).

La lógica interna verifica que:

- Haya conexión a internet.
- Existan registros pendientes (sincronizado = 0) en la base local.

12.3.2. Flujo técnico de sincronización

1. Se accede a la tabla SQLite del módulo (por ejemplo, tamizaje_visual).

2. Se recuperan todos los registros con sincronizado = 0.
3. Por cada registro:
    - Se envía mediante una petición HTTP POST al endpoint del backend:

POST /tamizaje_visual/sincronizar

Se incluye el token JWT en los headers para autenticar la petición:

Authorization: Bearer eyJhbGci...

Si el servidor responde con código 200 OK, el registro se actualiza localmente:

actualizarEstado(idLocal, 1); // sincronizado = 1

Si hay un error (por ejemplo, error 500 o sin conexión), el registro permanece intacto en SQLite para reintento posterior.

12.3.3. Estructura de la petición

El cuerpo (body) de la petición se arma en formato JSON desde Flutter. Por ejemplo, para tamizaje visual:

```
{
  "encuesta_id": "2ac3e1f9-f675-4304-9e38-...",
  "ojo_izquierdo_sin_lentes": "20/40",
  "ojo_derecho_sin_lentes": "20/30",
  "resultado_sin_lentes": "CON ERROR DE REFRACCIÓN"
}
```

Esto se envía utilizando la clase http de Flutter:

```
final response = await http.post(
  Uri.parse('$API_URL/tamizaje_visual/sincronizar'),
  headers: {
    'Content-Type': 'application/json',
    'Authorization': 'Bearer $token',
  },
```

```
  body: jsonEncode(registro),
);
```

12.3.4. Resultados esperados

La app responde con un mensaje tipo SnackBar:

☑ “Tamizaje sincronizado correctamente”

❌ “Error al sincronizar, intente más tarde”

Se evita duplicar registros en el servidor porque la API verifica si esa encuesta_id ya tiene un módulo registrado antes de crear o actualizar.

12.3.5. Ventajas del enfoque modular

- Permite sincronizar módulo por módulo incluso si solo uno fue completado.
- Si un módulo falla, los demás pueden continuar.
- Compatible con múltiples encuestadores trabajando al mismo tiempo.
- Escalable para nuevos módulos de salud.

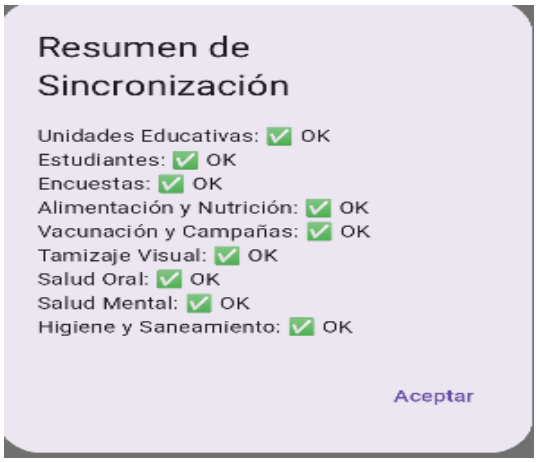


Figura 6. Registro de sincronización exitosa

12.4. Lógica de visibilidad por tipo de usuario

El sistema implementa una lógica de control de visibilidad que determina qué módulos puede ver o acceder un usuario, en función de:

- Su rol (admin, encuestador, odontologo, rector)
- El tipo de encuesta asociada (salud o salud_oral)

Esto permite que cada usuario vea solo las secciones del sistema que le corresponden, lo cual mejora la experiencia, previene errores y refuerza la seguridad de los datos

12.4.1. Control basado en roles

Al iniciar sesión en la app móvil, el backend devuelve un token JWT que contiene información del usuario, incluyendo su rol. Este rol se decodifica y se almacena localmente (por ejemplo, en SharedPreferences o en una variable global).

Según el rol, se define lo siguiente:

Rol	Acceso permitido
encuestador	Todos los módulos excepto Salud Oral
odontologo	Solo módulo de Salud Oral
admin	Acceso total (si se habilita en producción)
rector	No accede a app móvil (solo usa plataforma web)

12.4.2. Control basado en tipo de encuesta

Cada estudiante tiene una encuesta asignada con un campo tipo_encuesta, que puede ser:

- "salud" → Permite registrar nutrición, visual, mental, vacunación, higiene
- "salud_oral" → Solo permite salud oral

Al cargar la información del estudiante en la app, se verifica este campo para decidir qué módulos habilitar visualmente en la pantalla principal (ModuloSaludScreen).

12.4.3. Ejemplo técnico en Flutter

```
if (tipoEncuesta == 'salud') {

  // Mostrar todos los módulos excepto salud oral

} else if (tipoEncuesta == 'salud_oral') {

  // Mostrar solo módulo de salud oral

}
```

Y además:

```
if (rol == 'odontologo') {

  // Mostrar solo botón de Salud Oral

} else if (rol == 'encuestador') {

  // Mostrar todos menos oral

}
```

Esto se aplica dinámicamente al construir los botones del menú de módulos.

12.4.4. Beneficios de esta lógica

- Simplifica la interfaz para cada usuario.
- Minimiza errores al evitar registros no permitidos.
- Mejora el rendimiento visual en dispositivos con poca capacidad.
- Aumenta la seguridad y confidencialidad de los datos.
- 12.5. Funcionalidades clave de la aplicación móvil
- La app móvil desarrollada en Flutter concentra diversas funcionalidades integradas que permiten a los usuarios (encuestadores, odontólogos) registrar datos de forma eficiente y segura, trabajar sin conexión y sincronizar su trabajo con el servidor central. A continuación, se detallan las características principales:

12.5.1. Autenticación segura

- La app solicita cédula y contraseña en el inicio de sesión.
- Se valida contra el backend mediante una petición POST /login.
- Si las credenciales son válidas, el backend retorna un token JWT, que se almacena localmente.
- Todas las peticiones futuras a la API incluyen ese token para mantener la sesión activa.

12.5.2. Consulta y carga del estudiante

- La app permite buscar estudiantes por cédula.
- Muestra su información básica: nombres, curso, edad, sexo, tipo de encuesta.
- Calcula automáticamente la edad a partir de la fecha de nacimiento, con formato en años completos.

12.5.3. Formularios por módulo

- Cada módulo de salud cuenta con su propio formulario validado.
- Se utilizan componentes estándar como TextFormField, DropdownButton, Checkbox, DatePicker, etc.
- Se incluyen validaciones automáticas por campo (no vacío, tipo de dato correcto, rangos válidos).
- Algunos módulos realizan cálculos internos (IMC, conteo de piezas, clasificación visual, etc.).

12.5.4. Almacenamiento local en SQLite

- Cada formulario, al ser completado, guarda los datos en la base de datos SQLite.
- Se marca el registro con sincronizado = 0 para posterior envío.
- La información se mantiene en el dispositivo incluso si se cierra la app.

12.5.5. Sincronización automática

- Cuando hay conexión a internet, los registros pendientes se envían al backend.
- Se sincroniza módulo por módulo, sin afectar los ya registrados.
- Al recibir confirmación del servidor, el campo sincronizado se actualiza a 1.

12.5.6. Control de visibilidad por rol y tipo de encuesta

- Según el rol (encuestador, odontologo, etc.), se muestran o esconden módulos en el menú principal.
- Según el tipo de encuesta (salud, salud_oral), se activa solo lo que corresponde.

12.5.7. Interfaz intuitiva y responsiva

- Se utilizan Card, ListTile y botones estilizados (ElevatedButton) para facilitar el uso.
- Los formularios están organizados de forma clara, con títulos, subtítulos y secciones diferenciadas.
- Los mensajes de éxito y error se muestran mediante SnackBar con colores e íconos.

12.5.8. Seguridad y restricciones

- La app no permite editar datos ya sincronizados.
- No permite eliminar registros desde la interfaz.
- No se permite exportar datos desde la app, solo consultar o registrar.

13. PLataforma web administrativa (React)

La plataforma web del sistema fue desarrollada utilizando React.js, con enfoque modular, diseño profesional y conexión directa con la base de datos del sistema de salud mediante API REST. Está orientada a usuarios administradores y rectores, quienes pueden consultar registros, aplicar filtros, visualizar estadísticas de salud y gestionar usuarios del sistema.

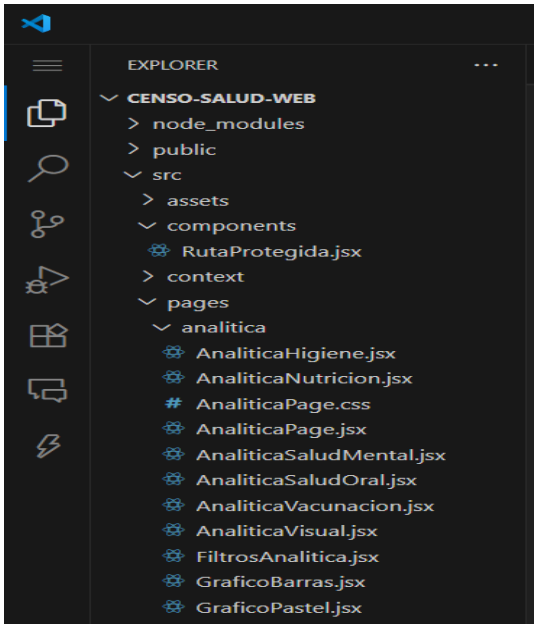
13.1. Estructura del proyecto React

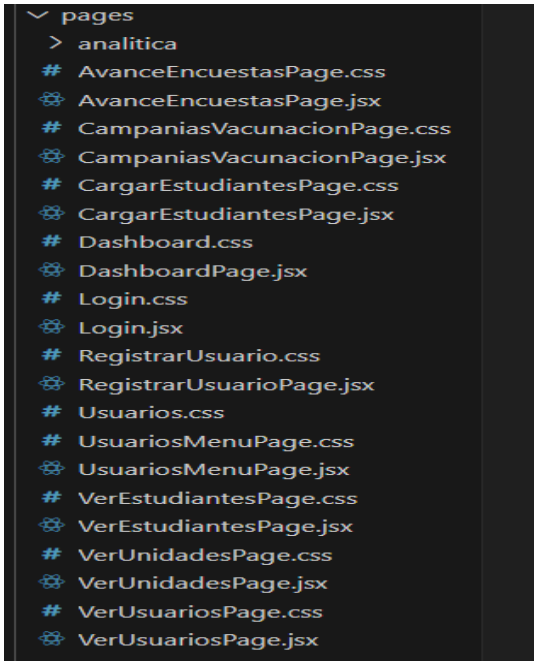
El proyecto sigue una arquitectura clara y escalable basada en carpetas modulares. La organización general del código fuente es la siguiente:

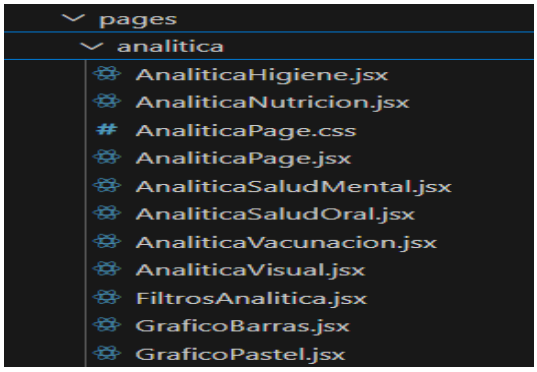
🗀 src/ contiene todos los archivos relevantes de la aplicación:

```
src/
├── assets/            # Imágenes y recursos estáticos
├── components/        # Componentes reutilizables como RutaProtegida
├── context/           # Manejo de sesiones
├── pages/             # Todas las pantallas del sistema divididas por secciones
│   ├── analitica/     # Páginas de analítica por módulo
│   ├── usuarios/      # Gestión de usuarios
│   └── estudiantes/   # Consulta y visualización de estudiantes
├── services/          # Lógica para llamar a la API (axios)
├── utils/             # Funciones auxiliares y archivos de configuración visual
└── App.jsx            # Navegación principal (ruteo)
```

├── config.js # Dirección base del backend
└── main.jsx # Archivo de entrada







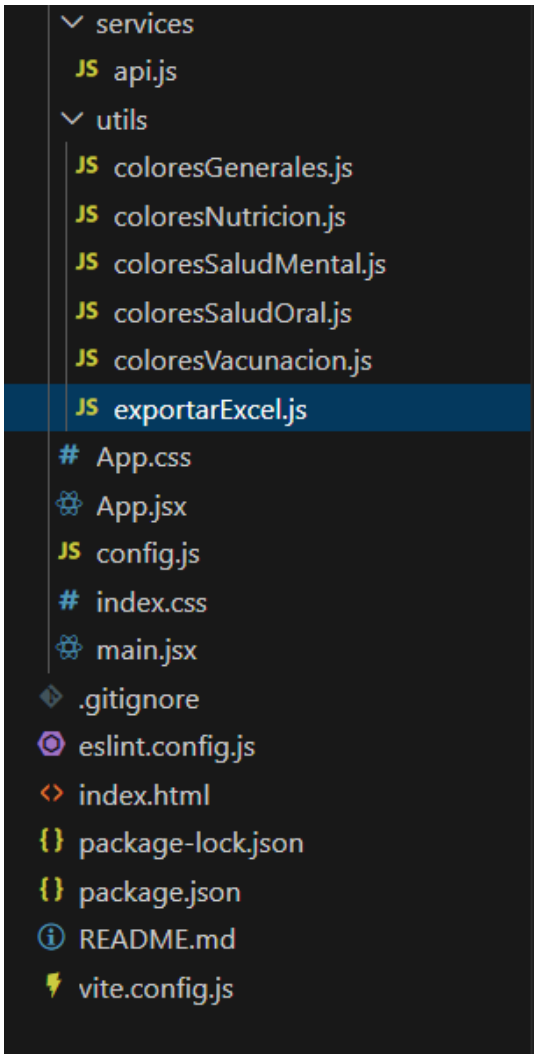


Figura 8. Estructura de carpetas del proyecto React (Visual Studio Code)

13.2. Inicio de sesión y control de sesión

- El login solicita cédula y contraseña.
- Tras validar las credenciales, el backend responde con un token JWT.
- El token se guarda en localStorage.
- Para proteger las rutas se utiliza el componente RutaProtegida.jsx, que evita el acceso a páginas si el usuario no está autenticado.

13.3. Gestión de usuarios

Accedida por el rol admin, esta sección incluye:

- Página de menú (UsuariosMenuPage.jsx) con acceso a registrar, ver y editar usuarios.
- Registro de usuarios con validación de campos (RegistrarUsuarioPage.jsx).
- Visualización y edición en tabla (VerUsuariosPage.jsx), con:
    - Filtros de búsqueda
    - Cambiar estado (Activo/Inactivo) mediante switch

- Botón para editar datos del usuario seleccionado

13.4. Consulta de estudiantes

- Página VerEstudiantesPage.jsx
- Permite filtrar por:
  - Unidad educativa
  - Curso
  - Cédula del estudiante
- Muestra los módulos de salud registrados (oral, nutrición, visual, etc.)
- Al hacer clic, se visualiza la información registrada por módulo
- Los datos no son editables desde la web (modo solo lectura)

13.5. Módulo de analítica

El módulo de analítica es uno de los componentes más relevantes del sistema. Está implementado mediante las siguientes páginas:

- AnaliticaPage.jsx: contenedor general que renderiza filtros y los módulos correspondientes
- FiltrosAnalitica.jsx: selector de unidad educativa, curso, sexo, edad y fecha
- AnaliticaNutricion.jsx, AnaliticaSaludOral.jsx, etc.: componentes para cada módulo
- GraficoBarras.jsx y GraficoPastel.jsx: componentes visuales que usan Recharts

Cada módulo puede:

- Aplicar filtros en tiempo real
- Mostrar gráficos estadísticos por categoría
- Exportar a Excel los registros fuera del rango normal

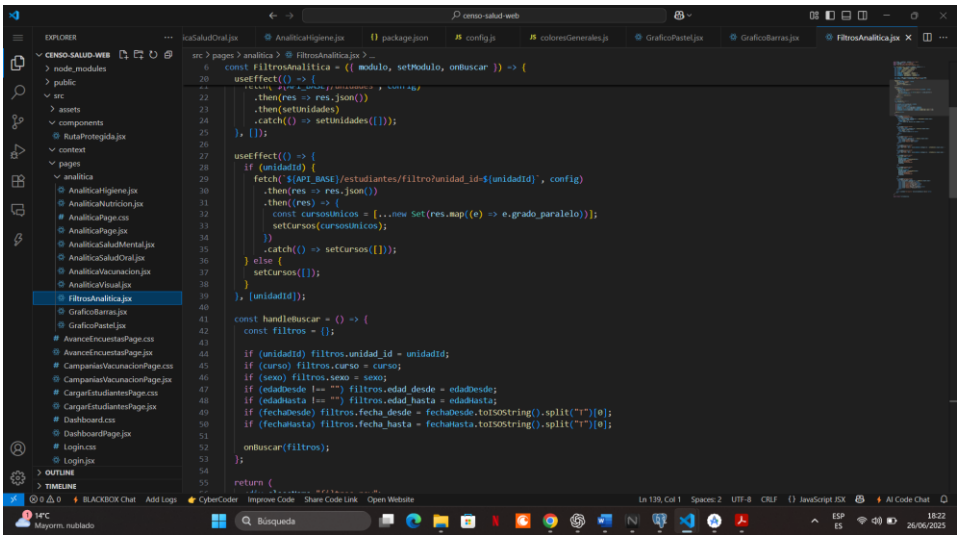


Figura 10. Módulo de analítica con filtros y gráficos por módulo de salud

13.6. Exportación de datos

Implementado en exportarExcel.js con uso de las librerías:

- xlsx: para generar archivos Excel
- file-saver: para permitir la descarga directa

Características:

- Cada módulo tiene un botón para exportar casos anormales
- El archivo se genera con nombre y columnas descriptivas
- Exporta los datos filtrados con todos los campos clave

Ejemplo del código de export.js

```
import axios from "axios";

import * as XLSX from "xlsx";

import { saveAs } from "file-saver";

import { toast } from "react-toastify";

import { API_BASE } from "../config";

const exportarAExcel = async ({ tipo, titulo, filtros, token }) => {

  try {

    let endpoint = "";

    if (tipo === "visual") {

      endpoint = `${API_BASE}/analitica/tamizaje/detalle`;

    } else if (["imc", "peso", "talla"].includes(tipo)) {

      endpoint = `${API_BASE}/analitica/nutricion/detalle`;

    } else if (tipo === "vacunacion-regular") {

      endpoint = `${API_BASE}/analitica/vacunacion/regular/detalle`;

    } else if (tipo === "vacunacion-campania") {

      endpoint = `${API_BASE}/analitica/vacunacion/campania/detalle`;

    } else if (tipo === "salud-oral-morbilidad") {

      endpoint = `${API_BASE}/analitica/salud-oral/detalle-morbilidad`;
```

```
} else if (tipo === "salud-oral-prevencion") {
  endpoint = `${API_BASE}/analitica/salud-oral/detalle-prevencion`;
} else if (["violencia", "consumo", "emocional"].includes(tipo)) {
  endpoint = `${API_BASE}/analitica/salud-mental/detalle`;
} else if (tipo === "higiene") {
  endpoint = `${API_BASE}/analitica/higiene/detalle`;
} else {
  toast.error("❌ Tipo de reporte no soportado.");
  return;
}
const esNutricion = ["imc", "peso", "talla"].includes(tipo);
const esSaludMental = ["violencia", "consumo", "emocional"].includes(tipo);
const params = esNutricion || esSaludMental ? { tipo, ...filtros } : filtros;
const res = await axios.get(endpoint, {
  headers: { Authorization: `Bearer ${token}` },
  params,
});
const datos = res.data;
if (!datos || datos.length === 0) {
  toast.warning(`⚠️ No hay datos para exportar en: ${titulo}`);
  return;
}
let datosFormateados = [];

if (tipo === "visual") {
```

```
  datosFormateados = datos.map((r) => ({
    "Fecha de Encuesta": new Date(r.fecha_encuesta).toLocaleDateString("es-EC"),
    "Apellidos": r.apellidos,
    "Nombres": r.nombres,
    "Edad": r.edad,
    "Curso": r.curso,
    "Profesor/a": r.profesor,
    "Sin Lentes OD": r.od_sin_lentes,
    "Sin Lentes OI": r.oi_sin_lentes,
    "Con Lentes OD": r.od_con_lentes,
    "Con Lentes OI": r.oi_con_lentes,
    "Resultado": r.resultado,
  }));
} else if (esNutricion) {
  datosFormateados = datos.map((r) => ({
    "Fecha de Encuesta": new Date(r.fecha_encuesta).toLocaleDateString("es-EC"),
    "Apellidos": r.apellidos,
    "Nombres": r.nombres,
    "Edad": r.edad,
    "Curso": r.curso,
    "Profesor/a": r.profesor,
    "Resultado": r.resultado,
  }));
} else if (tipo.startsWith("vacunacion")) {
  datosFormateados = datos.map((r) => ({
```

```
    "Apellidos": r.apellidos,
    "Nombres": r.nombres,
    "Cédula": r.cedula,
    "Edad": r.edad,
    "Curso": r.curso,
    "Profesor/a": r.profesor,
    "Vacuna": r.vacuna,
    "Estado": r.estado,
    "Fecha": r.fecha ? new Date(r.fecha).toLocaleDateString("es-EC") : "Sin fecha",
  }));
} else if (tipo === "salud-oral-morbilidad") {
  datosFormateados = datos.map((r) => ({
    "Fecha de Encuesta": new Date(r.fecha_encuesta).toLocaleDateString("es-EC"),
    "Apellidos": r.apellidos,
    "Nombres": r.nombres,
    "Edad": r.edad,
    "Curso": r.curso,
    "Profesor/a": r.profesor,
    "Cariadas (Sup)": r.cariadas_sup,
    "Cariadas (Inf)": r.cariadas_inf,
    "Obturadas (Sup)": r.obturadas_sup,
    "Obturadas (Inf)": r.obturadas_inf,
    "Perdidas (Sup)": r.perdidas_sup,
    "Perdidas (Inf)": r.perdidas_inf,
    "Sanas (Sup)": r.sanas_sup,
```

```
    "Sanas (Inf)": r.sanas_inf,
    "Total Piezas": r.total_piezas,
    "No Aplica": r.piezas_no_aplica,
  }));
} else if (tipo === "salud-oral-prevencion") {
  datosFormateados = datos.map((r) => ({
    "Fecha de Encuesta": new Date(r.fecha_encuesta).toLocaleDateString("es-EC"),
    "Apellidos": r.apellidos,
    "Nombres": r.nombres,
    "Edad": r.edad,
    "Curso": r.curso,
    "Profesor/a": r.profesor,
    "Placa Bacteriana / Profilaxis": r.placa_bacteriana === "SÍ" ? r.fecha_placa : "NO",
    "Necesita Flúor": r.fluor === "SÍ" ? r.fecha_fluor : "NO",
    "Necesita Sellantes": r.sellantes === "SÍ" ? r.fecha_sellantes : "NO",
  }));
} else if (esSaludMental) {
  datosFormateados = datos.map((r) => ({
    "Fecha de Encuesta": new Date(r.fecha_encuesta).toLocaleDateString("es-EC"),
    "Apellidos": r.apellidos,
    "Nombres": r.nombres,
    "Edad": r.edad,
    "Curso": r.curso,
    "Profesor/a": r.profesor,
    "Resultado": r.resultado,
```

```
  }));
} else if (tipo === "higiene") {
  datosFormateados = datos.map((r) => {
    const fila = {
      "Fecha de Encuesta": r["Fecha de Encuesta"],
      "Apellidos": r["Apellidos"],
      "Nombres": r["Nombres"],
      "Sexo": r["Sexo"],
      "Edad": r["Edad"],
      "Curso": r["Curso"],
      "Profesor/a": r["Profesor"],
    };
    if (r["¿Participa en eliminación de riesgos de vectores?"] === "NO")
      fila["¿Participa en eliminación de riesgos de vectores?"] = "NO";
    if (r["¿Participa en espacios saludables?"] === "NO")
      fila["¿Participa en espacios saludables?"] = "NO";
    if (r["¿Participa en lavado de manos?"] === "NO")
      fila["¿Participa en lavado de manos?"] = "NO";
    if (r["¿Participa en consumo de agua segura?"] === "NO")
      fila["¿Participa en consumo de agua segura?"] = "NO";
    return fila;
  });
}
const ws = XLSX.utils.json_to_sheet(datosFormateados, { origin: "A1" });
```

```
    const cabeceras = Object.keys(datosFormateados[0]);
    cabeceras.forEach((key, i) => {
      const col = String.fromCharCode(65 + i) + "1";
      if (ws[col]) {
        ws[col].s = {
          font: { bold: true },
          alignment: { horizontal: "center" },
          fill: { fgColor: { rgb: "E8EAF6" } },
        };
      }
    });
    ws["!cols"] = cabeceras.map(() => ({ wch: 22 }));
    const wb = XLSX.utils.book_new();
    XLSX.utils.book_append_sheet(wb, ws, "Reporte");
    const nombreArchivo = `Reporte_${titulo.replace(/\s/g, "_")}.xlsx`;
    const excelBuffer = XLSX.write(wb, { bookType: "xlsx", type: "array" });
    const blob = new Blob([excelBuffer], { type: "application/octet-stream" });
    saveAs(blob, nombreArchivo);
  } catch (error) {
    console.error("❌ Error al exportar:", error.message);
    toast.error("❌ Error al exportar el reporte.");
  }
};
export default exportarAExcel;
```

13.7. Estilo visual e institucional

- Diseño moderno, claro y con colores institucionales del Ministerio de Salud
- Tipografía legible, distribución en tarjetas y gráficos amigables
- Logo institucional en el encabezado y botones bien distribuidos

13.8. Seguridad

- Autenticación protegida con JWT
- Verificación de rol antes de acceder a ciertas funciones
- Datos sensibles de los estudiantes no pueden ser modificados desde la web

14. INSTALACIÓN DEL SISTEMA

El sistema desarrollado está compuesto por tres partes principales:

- Un backend (API REST) desarrollado en Node.js con conexión a PostgreSQL.
- Una aplicación móvil desarrollada en Flutter, capaz de operar offline.
- Una plataforma web desarrollada en React.js, orientada a perfiles administrativos y de consulta.

14.1 Almacenamiento del Proyecto

- El sistema fue **almacenado localmente** durante todo el proceso de desarrollo, utilizando Visual Studio Code como entorno principal. No se utilizó un repositorio Git remoto como GitHub o GitLab.

Módulo	Lenguaje/Framework	Carpeta local
Backend API	Node.js + PostgreSQL	BACKEND-CENSO-NODE
App móvil	Flutter	censo_salud
Plataforma web	React.js	CENSO-SALUD-WEB

📌 Se recomienda en futuros despliegues incorporar el uso de Git y almacenamiento remoto para facilitar la colaboración, el versionado y los respaldos.

14.1 Instalación del backend (Node.js + PostgreSQL)

14.1.1 Requisitos previos

- Node.js v18 o superior
- PostgreSQL v15 o superior
- NPM (Node Package Manager)
- pgAdmin o cliente SQL

14.1.2 Estructura local

El proyecto backend se encuentra en la carpeta:

BACKEND-CENSO-NUEVO

Contiene los siguientes archivos y directorios relevantes:

- /routes: todas las rutas por módulo (alimentación, encuestas, estudiantes, etc.)
- /middlewares: validación de tokens y roles
- db.js: conexión con la base de datos PostgreSQL
- server.js: archivo principal del servidor

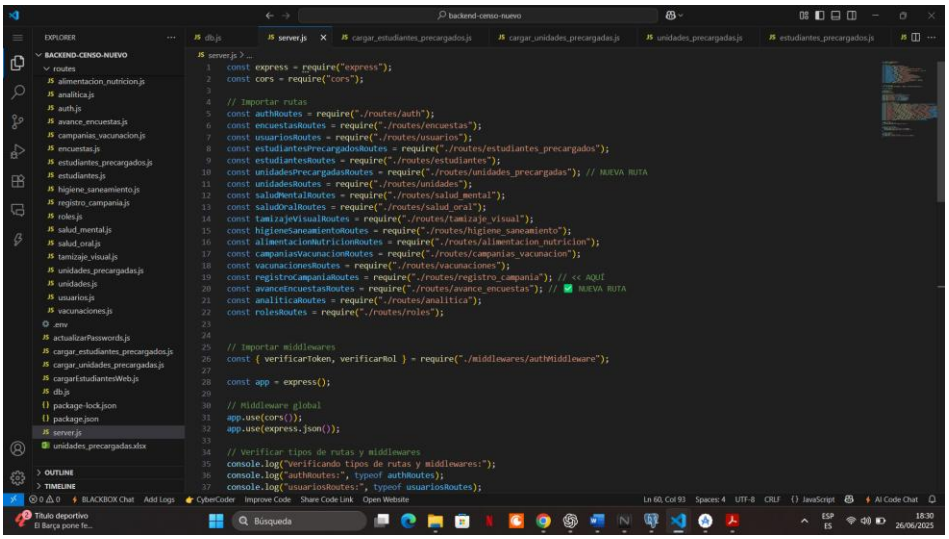


Figura 11. Vista del backend en VSCode

14.1.3 Instalación y ejecución

1. Ingresar a la carpeta del backend:

```
bash

cd BACKEND-CENSO-NODE
```

Instalar dependencias:

```
bash

npm install
```

3. Configurar archivo .env con los datos de conexión:

```env
PORT=5000
DB_USER=postgres
DB_PASSWORD=tu_clave
DB_NAME=censo_salud
DB_HOST=localhost
DB_PORT=5432
JWT_SECRET=clave_segura
```

Crear la base de datos desde PostgreSQL:

```sql
CREATE DATABASE censo_salud;
```

Importar todas las tablas (ver diagrama y diccionario técnico).

Ejecutar el backend:

```bash
node server.js
```

El backend quedará escuchando en el puerto 5000.

14.2 Instalación de la Aplicación Móvil (Flutter)

14.2.1 Requisitos

- Flutter SDK instalado
- Android Studio o Visual Studio Code
- Emulador o dispositivo Android

14.2.2 Estructura

Ubicada en:

```bash
censo_salud/
```

Contiene las carpetas lib/models, lib/database, lib/screens, y lib/services.

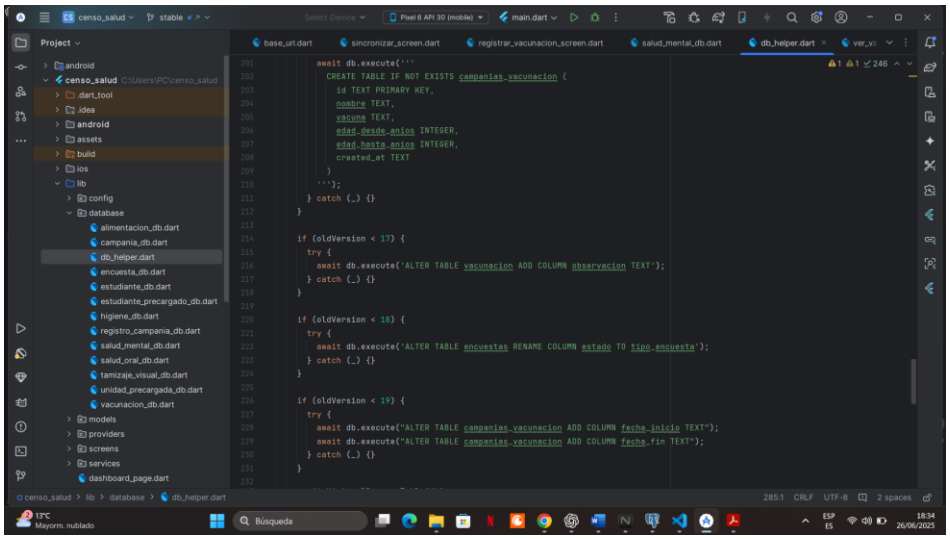


Figura 12. Vista de carpetas del proyecto Flutter

14.2.3 Instalación y ejecución

1. Ingresar al directorio:

```bash
cd censo_salud
```

Descargar dependencias:

```bash
flutter pub get
```

⚠ La app funciona con almacenamiento local SQLite y sincroniza datos con el backend cuando hay conexión disponible.

14.3 Instalación de la Plataforma Web (React.js)

14.3.1 Requisitos

- Node.js y npm instalados
- Navegador actualizado

14.3.2 Estructura

El proyecto está en:

```bash
CENSO-SALUD-WEB/
```

Contiene subcarpetas como src/pages, src/utils, src/components, src/services.

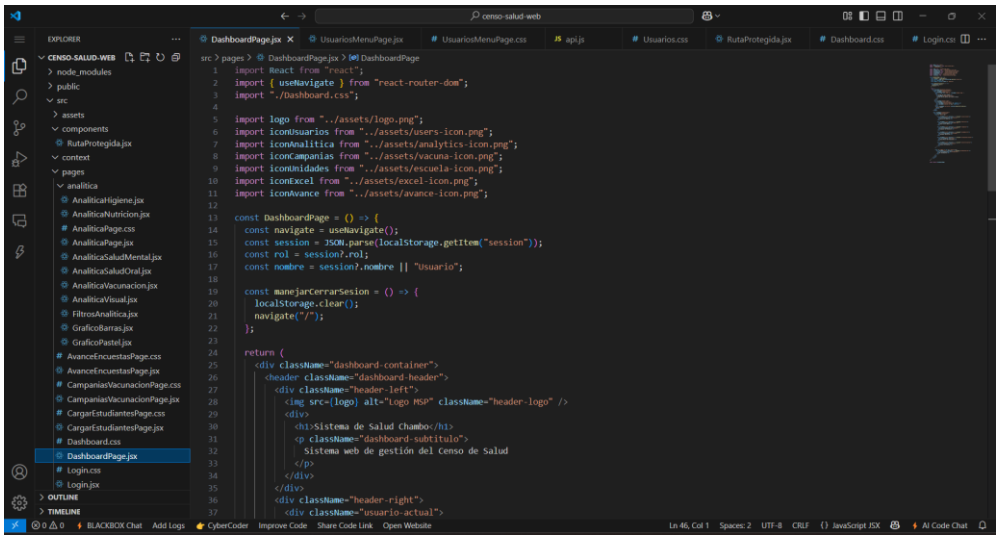


Figura 13. Estructura del frontend React

14.3.3 Instalación y ejecución

1. Ingresar al directorio:

```bash
cd CENSO-SALUD-WEB
```

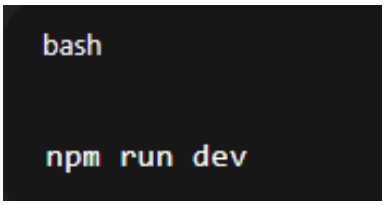
Instalar dependencias:

```bash
npm install
```

Configurar conexión al backend en src/config.js:

```js
export const API_URL = "http://localhost:5000";
```

Ejecutar la app:



La app web estará disponible en:

http://localhost:5173

14.4 Recomendaciones de instalación en producción

- Backend: servidor Node (ej. VPS, Railway, Render)
- Base de datos: PostgreSQL en servidor seguro
- Web: Vercel, Netlify o servidor Apache/Nginx
- App móvil: compilar APK para distribución

15. Mantenimiento y futuros ajustes

El sistema de gestión del Censo de Salud fue diseñado con una arquitectura modular y tecnologías escalables, lo cual permite su mantenimiento, actualización y expansión de forma controlada y sostenible en el tiempo.

Esta sección describe las buenas prácticas, acciones recomendadas para el mantenimiento preventivo y correctivo, así como las pautas para incorporar nuevos módulos o extender el sistema hacia otros contextos geográficos o funcionales.

15.1. Buenas prácticas de mantenimiento

- **Backups regulares de la base de datos**
  - Programar respaldos automáticos diarios o semanales.
  - Guardar copias en ubicaciones seguras y externas al servidor principal.
- **Actualización de dependencias**
  - Revisar periódicamente versiones de:
    - Node.js
    - NPM packages (backend y frontend)
    - Flutter SDK
  - Ejecutar npm outdated o flutter upgrade para mantener componentes actualizados y seguros.
- **Control de acceso y roles**
  - Revisar usuarios activos periódicamente.
  - Eliminar cuentas obsoletas o sin uso.
  - Cambiar contraseñas ante sospecha de incidentes.

- **Monitoreo de logs y errores**
    - Configurar logs de servidor para capturar errores.
    - Revisar logs ante comportamientos inusuales.
    - Aplicar correcciones preventivas.

15.2. Procedimientos ante errores comunes

- **Error de conexión al backend**
    - Revisar archivo .env o config.js.
    - Verificar puerto de backend y disponibilidad del servidor.
- **Problemas de sincronización en app móvil**
    - Verificar conectividad del dispositivo.
    - Comprobar si el backend está corriendo.
    - Revisar logs de errores en Flutter.
- **Errores en consultas de la web**
    - Revisar token JWT en localStorage.
    - Verificar expiración de sesión.
    - Consultar consola del navegador para mensajes de error.

15.3. Cómo agregar un nuevo módulo de salud

Gracias a su diseño modular, el sistema permite incorporar nuevos módulos siguiendo los pasos:

15.3.1. En la base de datos (PostgreSQL)

- Crear una nueva tabla que relacione su encuesta_id con los datos específicos del módulo.
- Definir claves primarias UUID.
- Agregar campos created_at y updated_at si se requiere trazabilidad.

15.3.2. En el backend (Node.js)

- Crear archivo nuevo en /routes/ (ej. nuevomodulo.js).
- Crear controlador en /controllers/ para manejar la lógica CRUD y sincronización.
- Configurar validaciones específicas.
- Definir endpoint /nuevomodulo/sincronizar.

15.3.3. En la app móvil (Flutter)

- Crear modelo en lib/models/.
- Implementar nueva pantalla en lib/screens/.
- Crear métodos para guardar en SQLite y sincronizar.
- Incluir nuevo botón en ModuloSaludScreen si el rol lo permite.

15.3.4. En la web (React)

- Crear página en src/pages/ para consulta y visualización.
- Implementar servicio en src/services/.
- Integrar en el dashboard o analítica.
- Opcional: agregar gráficos en Recharts.

15.4. Escalabilidad futura

El sistema puede escalar en distintas dimensiones:

- **Cobertura geográfica**
  - Incorporar nuevas provincias o cantones.
  - Replicar estructura de unidades educativas en otras zonas.
- **Funcionalidad**
  - Agregar nuevos módulos clínicos según necesidades del MSP.
  - Integrar analítica avanzada con inteligencia artificial o machine learning.
- **Infraestructura**
  - Migrar backend y base de datos a servidores cloud escalables.
  - Implementar balanceo de carga en caso de alto tráfico.

15.5. Recomendaciones finales

- Mantener documentación técnica actualizada.
- Aplicar pruebas periódicas de carga y seguridad.
- Planificar sesiones de capacitación para nuevos usuarios.
- Documentar cambios en versiones y mantener historial de actualizaciones.