

UNIVERSIDAD NACIONAL DE INGENIERÍA

Facultad de Ingeniería Industrial y de Sistemas



SI807U Sistemas de Inteligencia de Negocios

DOCENTE:

Dr. Ing Aradiel Castañeda, Hilario

INTEGRANTES:

Código UNI	Apellidos y Nombres	Correo Electrónico	Tareas realizadas
20220008E	Andrade Saavedra, Navhi Giordano	navhi.andrade.s@uni.pe	•
20220122B	Caruzo Cieza, David	david.caruzo.c@uni.pe	•
20200298H	Carhuas Romero Jhon Jesus	jhon.carhuas.r@uni.pe	•

Grupo N°2

2025-II

INDICE

INDICE DE FIGURAS

INDICE DE TABLAS

DESCRIPCIÓN GENERAL DE LA EMPRESA

1.1. Nombre / Razón social

Razón social: Seguro Social de Salud (EsSalud)

RUC: 20131257750

1.2. Giro de la empresa

Es un organismo público descentralizado del Estado peruano, con autonomía técnica, administrativa, económica, financiera, presupuestal y contable.

Su función principal es brindar servicios de seguridad social en salud, es decir:

- Prestaciones médicas: promoción, prevención, diagnóstico, tratamiento, rehabilitación.
- Prestaciones económicas y sociales vinculadas a riesgos humanos.

1.3. Ubicación(es) y presencia geográfica

Su sede principal está en Lima, distrito de Jesús María, pero tiene red de establecimientos de salud distribuidos por todo el Perú: cobertura nacional.

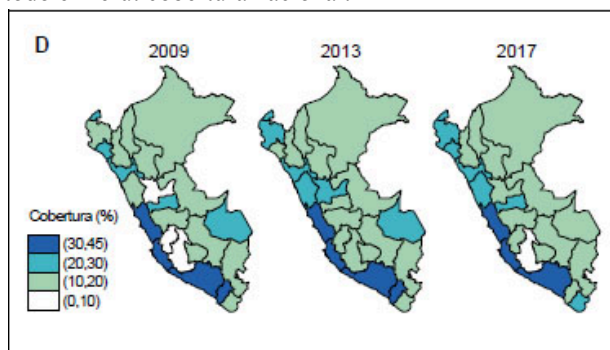


Figura X. Cobertura - EsSalud

En total, posee cerca de 400 establecimientos (hospitales generales, policlínicos, especializados) en diferentes regiones del país.

Además, forma parte del sistema estatal: sus líneas de coordinación institucional la conectan con el Ministerio de Salud, el Ministerio de Economía, y otros entes públicos.

1.4. Misión y visión

Misión

“Brindamos prestaciones de salud, económicas y sociales a nuestros asegurados con una gestión eficiente e innovadora que garantiza la protección financiera de las prestaciones integrales.”

Visión

“Ser una institución moderna y en mejora continua, centrada en los asegurados, que garantiza el acceso a la seguridad social en salud con ética, oportunidad y calidad.”

1.5. Valores y objetivos estratégicos

Valores institucionales

Según la entidad y su normativa interna (Código de Ética, manuales institucionales), algunos valores y principios que rigen la acción de EsSalud son:

- Solidaridad
- Universalidad
- Igualdad
- Integralidad
- Ética institucional

El Código de Ética / Código de Conducta de EsSalud recoge formalmente esos valores, principios, deberes, prohibiciones y el modelo de conducta esperada para sus funcionarios.

Objetivos estratégicos institucionales

Del Plan Estratégico Institucional 2025-2030 y documentos relacionados, se pueden extraer algunos objetivos estratégicos clave que orientan la gestión de EsSalud:

Objetivo estratégico	Detalle / énfasis
Mejorar la calidad de los servicios	Elevar estándares, reducir errores, mejorar atención al asegurado
Garantizar el acceso efectivo	Promover cobertura real, reducir barreras geográficas o económicas
Transformación institucional	Modernización, innovación en modelos de gestión
Eficiencia y modernización de la gestión	Optimización de recursos, procesos internos ágiles
Gestión ética e integridad institucional	Fortalecer controles internos, transparencia, responsabilidad
Innovación tecnológica en salud	Introducir soluciones digitales, infraestructura tecnológica

ANÁLISIS ORGANIZACIONAL

2.1. Organigrama y roles clave

Organigrama

El organigrama de EsSalud muestra una estructura jerárquica amplia y especializada que integra órganos de alta dirección, gerencias centrales, oficinas y subgerencias distribuidas por funciones estratégicas, asistenciales y de apoyo. Esta organización busca garantizar una gestión coordinada de los servicios de salud, aseguramiento, información y planificación institucional a nivel nacional.

Nivel / Unidad	Rol Clave	Función dentro del caso (centrada en la gestión y uso de información)
Presidencia Ejecutiva / Gerencia General	Patrocinador Institucional	Autoriza y respalda la iniciativa de análisis predictivo como parte de la estrategia nacional de prevención de enfermedades crónicas. Facilita la coordinación intergerencial.
Gerencia Central de Planeamiento y Presupuesto	Gestor Estratégico de Indicadores	Define los objetivos de salud institucional y establece los KPI para medir el impacto de la prevención (reducción de diagnósticos, mejora de cobertura, eficiencia regional).

Gerencia de Gestión de la Información	Coordinador de Integración y Calidad de Datos	Reúne, valida y estandariza los datos provenientes de hospitales, asegurados y registros epidemiológicos para construir una base confiable de análisis.
Gerencia Central de Prestaciones de Salud	Unidad Usaria Principal del Modelo Predictivo	Utiliza los resultados del análisis para diseñar estrategias de intervención temprana, priorizar regiones y orientar campañas de salud preventiva.
Oficina de Inteligencia e Información Sanitaria	Analista Epidemiológico y Decisor Técnico	Interpreta los datos, construye mapas de riesgo y emite reportes para la toma de decisiones clínicas y operativas en función de los patrones detectados.
Gerencia de Atención Primaria / Subgerencia de Promoción de la Salud y Prevención de Enfermedades	Ejecutor de Intervenciones Preventivas	Implementa programas de educación, control y seguimiento en las zonas identificadas como de mayor riesgo por el modelo.
Gerencia Central de Seguros y Prestaciones Económicas	Proveedor de Datos Demográficos y de Cobertura	Facilita información de la población asegurada (edad, sexo, ubicación, tipo de seguro) para caracterizar los grupos vulnerables.
Gerencia Central de Operaciones	Coordinador Regional de Implementación	Supervisa la aplicación de acciones preventivas y verifica que las redes asistenciales actúen según las prioridades geográficas identificadas.
Gerencia Central de Gestión Financiera	Gestor de Recursos Preventivos	Asigna presupuesto a las regiones con mayor incidencia y prioriza la inversión en equipamiento, campañas y personal médico preventivo.
Gerencia Central de Gestión de las Personas	Gestor del Talento y Capacitación en Prevención	Promueve la formación del personal médico y técnico en manejo de datos epidemiológicos, detección temprana y educación sanitaria.

2.2. Productos y clientes

A. Productos

- **Prestaciones de Salud:** Comprenden los servicios médicos preventivos, asistenciales y de rehabilitación brindados en los diferentes niveles de atención, orientados a preservar y recuperar la salud del asegurado.
- **Prestaciones Económicas:** Incluyen subsidios por maternidad, lactancia, sepelio, incapacidad temporal y riesgos laborales, garantizando la protección financiera del asegurado.
- **Prestaciones Sociales:** Agrupan programas de bienestar social, atención al adulto mayor, rehabilitación física y laboral, y apoyo a personas con discapacidad.
- **Atención Primaria y Especializada:** Ofrecen consultas médicas, exámenes diagnósticos, hospitalización, telemedicina y servicios de alta complejidad en las redes asistenciales.
- **Programas Preventivos y de Promoción de la Salud:** Se enfocan en campañas de vacunación, detección temprana de enfermedades crónicas y educación sanitaria para fomentar hábitos saludables.
- **Gestión del Aseguramiento:** Comprende los procesos de registro, acreditación y control de cobertura de los asegurados y sus derechohabientes.

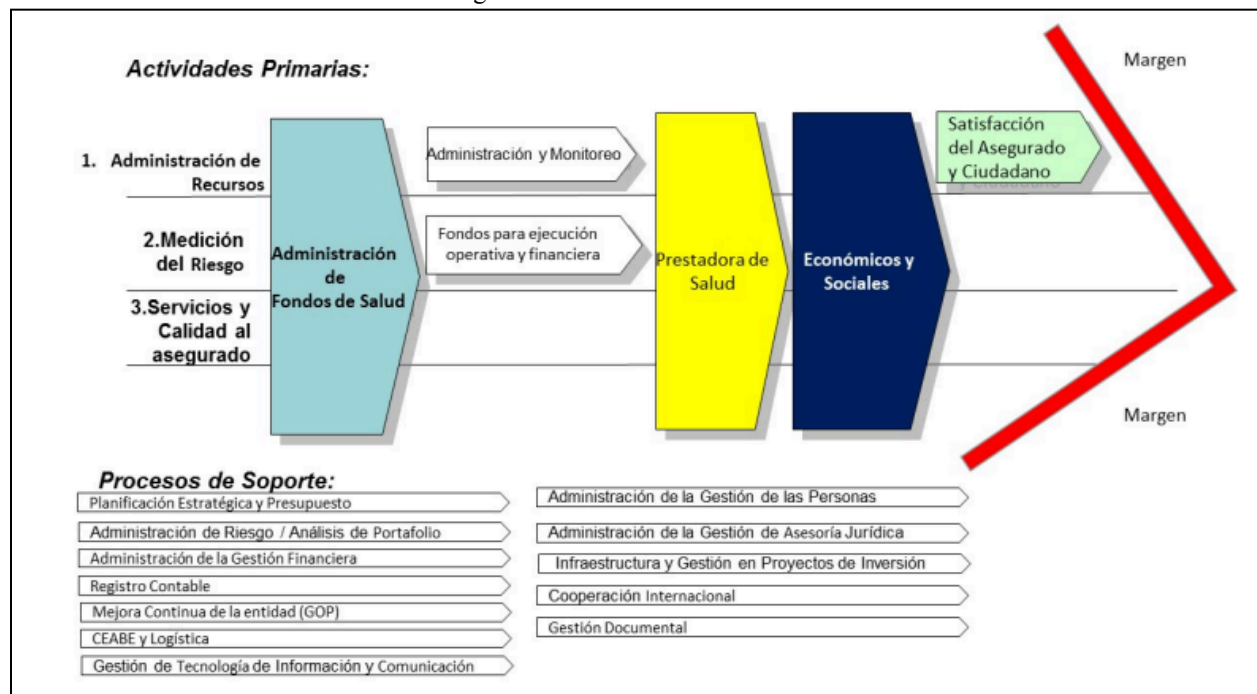
B. Clientes

- **Asegurados Regulares:** Trabajadores formales que aportan al sistema de seguridad social junto con sus derechohabientes, beneficiarios principales de los servicios de salud.
- **Pensionistas y Jubilados:** Personas retiradas del sistema laboral que mantienen acceso a los servicios médicos y económicos de EsSalud.
- **Trabajadores Independientes Asegurados:** Ciudadanos que realizan aportes voluntarios para acceder a las prestaciones de salud y protección económica.
- **Personas en Situación de Vulnerabilidad:** Adultos mayores, personas con discapacidad y grupos sociales que reciben atención prioritaria a través de programas sociales.
- **Empleadores y Empresas:** Instituciones públicas o privadas que contribuyen al financiamiento del seguro social y gestionan la afiliación de su personal.
- **Instituciones del Estado y Aliados Estratégicos:** Entidades gubernamentales que colaboran con EsSalud en campañas nacionales, convenios de cooperación y proyectos de salud pública.

2.3. Cadena de valor

La cadena de valor de EsSalud representa cómo la institución organiza y articula sus procesos primarios y de soporte para cumplir su misión de brindar prestaciones integrales de salud, económicas y sociales, orientadas a la satisfacción del asegurado y del ciudadano.

Figura :Cadena de valor EsSalud



Fuente: EsSalud

2.4. Procesos principales y secundarios

2.4.1 Actividades Primarias

- Administración de Recursos**
 - Comprende la planificación, asignación y gestión eficiente de los recursos económicos, humanos y materiales.
 - Asegura la disponibilidad de fondos para las operaciones institucionales y garantiza la sostenibilidad del sistema.
 - Se relaciona con la función de Administración y Monitoreo dentro de la gestión financiera.
- Medición del Riesgo**
 - Evalúa y controla los riesgos relacionados con la salud, la atención y la sostenibilidad financiera del sistema de seguridad social.
 - Incluye el análisis actuarial y de portafolio, permitiendo ajustar estrategias y fondos según el comportamiento de los asegurados.
 - Es la base para la Administración de Fondos de Salud.
- Servicios y Calidad al Asegurado**
 - Es la parte operativa y asistencial, donde se materializa la atención de salud.
 - A través de la Prestadora de Salud, se administran y ejecutan los fondos en forma de servicios médicos, preventivos y hospitalarios.
 - Busca garantizar eficiencia, oportunidad y calidad en la atención.

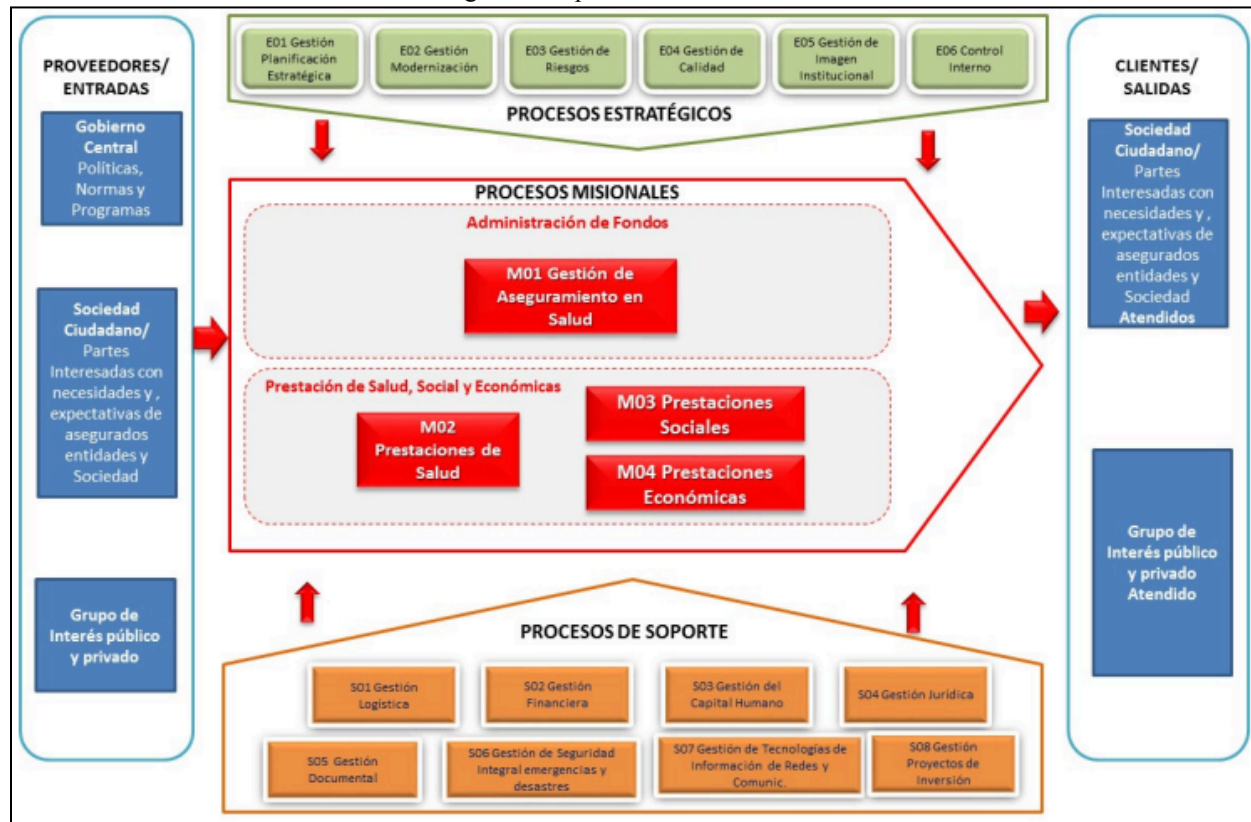
2.4.2 Actividades de Soporte

- Planificación Estratégica y Presupuesto:** Define las metas institucionales y la asignación de recursos.
- Administración de Riesgo / Gestión Financiera / Registro Contable:** Soportan el control interno y la transparencia económica.
- Gestión de Personas y Asesoría Jurídica:** Aseguran el desarrollo del talento humano y la legalidad de las operaciones.

- Gestión de Tecnología de la Información y Comunicación (TIC): Facilita la transformación digital, interoperabilidad y automatización de procesos.
- Infraestructura y Proyectos de Inversión: Garantizan la mejora continua de hospitales, redes y equipamiento médico.
- Gestión Documental y Logística: Apoyan la trazabilidad, el almacenamiento y la eficiencia administrativa.
- Cooperación Internacional y Mejora Continua: Promueven la innovación y el aprendizaje institucional permanente.

2.5. Mapa de procesos (diagramas)

Figura : Mapa de Proceso EsSalud



Fuente : EsSalud

Procesos

En EsSalud, los procesos representan el eje estructural que articula la gestión institucional y permite alinear la estrategia con la prestación efectiva de servicios de salud, sociales y económicos. Analizar estos procesos brinda una visión integral del funcionamiento del sistema de seguridad social, evidenciando cómo cada área contribuye al cumplimiento de la misión institucional de garantizar la protección y bienestar del asegurado. La cual se puede observar en el mapa de procesos y que organiza las actividades en tres niveles: estratégicos, misionales, soporte

A. Procesos Estratégicos

Son los que orientan, planifican y evalúan el funcionamiento global de la institución.

- **Gestión de Planificación Estratégica:** Define los objetivos, metas y planes institucionales alineados a las políticas nacionales de salud.
- **Gestión de Modernización:** Promueve la innovación, simplificación administrativa y mejora continua de los procesos institucionales.

- **Gestión de Riesgos:** Identifica, evalúa y mitiga los riesgos que pueden afectar la operatividad o sostenibilidad del sistema de salud.
- **Gestión de la Calidad:** Asegura el cumplimiento de estándares de calidad en los servicios médicos, administrativos y financieros.
- **Gestión de Imagen Institucional:** Fortalece la comunicación, transparencia y reputación de EsSalud ante la sociedad.
- **Control Interno:** Supervisa la correcta ejecución de los procesos y garantiza la integridad y legalidad de las operaciones.

B. Procesos Misionales

Constituyen el núcleo operativo de EsSalud, directamente relacionados con su razón de ser: brindar prestaciones integrales de salud, sociales y económicas.

- **Gestión de Aseguramiento en Salud:** Administra el registro, control y cobertura de los asegurados, asegurando su acceso a las prestaciones de salud.
- **Prestaciones de Salud:** Comprende la atención médica, preventiva y de rehabilitación en los diferentes niveles asistenciales.
- **Prestaciones Sociales:** Brinda servicios sociales de apoyo, bienestar y rehabilitación a los asegurados y sus familias.
- **Prestaciones Económicas:** Gestiona subsidios y compensaciones económicas (por maternidad, incapacidad, lactancia, sepelio, etc.).

C. Procesos de Soporte

Apoyan la ejecución de los procesos misionales, garantizando la disponibilidad de recursos, infraestructura y servicios administrativos.

- **Gestión Logística:** Administra la adquisición, almacenamiento y distribución de bienes y servicios.
- **Gestión Financiera:** Controla los recursos económicos, presupuestos y rendición de cuentas institucional.
- **Gestión del Capital Humano:** Se encarga de la selección, capacitación y bienestar del personal de EsSalud.
- **Gestión Jurídica:** Asesora y garantiza el cumplimiento de las normas legales y procedimientos administrativos.
- **Gestión Documental:** Asegura la organización, custodia y trazabilidad de la información institucional.
- **Gestión de Seguridad Integral, Emergencias y Desastres:** Implementa medidas de prevención, respuesta y continuidad operativa ante emergencias.
- **Gestión de Tecnologías de Información y Comunicación (TIC):** Desarrolla y mantiene los sistemas informáticos e infraestructura digital de EsSalud.
- **Gestión de Proyectos de Inversión:** Planifica y ejecuta proyectos de infraestructura, equipamiento y expansión de servicios.

2.2. Diagramas de flujo de procesos críticos

DIAGNÓSTICO Y DEFINICIÓN DE PROBLEMAS

3.1. Identificación de problemas de negocio

3.1.1 Análisis foda

Fortalezas	Oportunidades
------------	---------------

<p>Datos históricos institucionales de atenciones (emergencias, hospitalizaciones): Estos datos pueden ser la base para la construcción de modelos predictivos que permitan anticipar picos de demanda.</p> <p>Investigación previa en EsSalud sobre modelos predictivos para demanda en emergencias: Utilización de algoritmos de predicción que podrían integrarse con el análisis de la demanda futura.</p> <p>Procesos normados de distribución de camas hospitalarias: Manuales internos que, con ajustes, pueden optimizarse para prever la distribución de camas en función de la demanda anticipada.</p> <p>Soluciones digitales para el asegurado: Potencial para integrar plataformas de monitoreo de demanda y de gestión en tiempo real.</p> <p>Presencia de una red de establecimientos a nivel nacional: Ayuda a redistribuir la carga entre distintos centros, mejorando la capacidad de respuesta ante picos.</p> <p>Médicos reconocidos por su nivel de especialización y compromiso: Ayuda a optimizar los recursos humanos disponibles, lo que resulta clave ante incrementos de demanda.</p>	<p>Posibilidad de usar técnicas de series temporales y machine learning: Modelos como ARIMA/SARIMA para predecir picos de demanda, en especial en emergencias y hospitalización.</p> <p>Cooperación con entidades académicas: Generación de modelos de demanda oculta y proyectada, basados en estudios como el de estimación de demanda de servicios de salud.</p> <p>Impulso del gobierno para la transformación digital: Potencial para desarrollar herramientas predictivas a nivel nacional que ayuden a anticipar necesidades de infraestructura y recursos humanos.</p> <p>Crecimiento económico del país: Oportunidad de captar financiamiento o inversiones para fortalecer las infraestructuras y mejorar la capacidad de anticipación de demanda.</p> <p>Crecimiento de la población urbana: Aumenta la demanda de servicios médicos, lo que hace aún más urgente contar con una estrategia de anticipación de recursos.</p>
Debilidades	Amenazas
<p>Falta de sistematización e integración de datos históricos asistenciales: Los datos están disgregados y carecen de interoperabilidad, lo que dificulta su uso en modelos predictivos.</p> <p>Baja cultura institucional de planificación predictiva: Se enfoca más en respuestas reactivas ante la demanda que en la anticipación, lo que limita la capacidad de reacción ante picos inesperados.</p> <p>Recursos humanos rígidos: El personal no puede ajustarse rápidamente a aumentos en la demanda, lo que genera cuellos de botella y sobrecarga de trabajo.</p> <p>Procesos de asignación de camas no orientados a la anticipación: Los procedimientos actuales se basan en la distribución de camas de manera reactiva, sin contemplar una previsión a futuro que permita una mejor gestión ante aumentos repentinos de la demanda.</p> <p>Escasez presupuestal: La falta de recursos para mantener capacidad ociosa limita la posibilidad de</p>	<p>Eventos inesperados (epidemias, brotes, desastres naturales): Estos eventos pueden disparar de manera súbita la demanda de servicios de salud, lo que pone a prueba la capacidad de anticipación y respuesta.</p> <p>Incremento sostenido de demanda en emergencias: Ejemplo: en el Hospital Almenara, las esperas de hasta 96 horas por falta de camas revelan la incapacidad para manejar la demanda en picos.</p> <p>Violaciones de datos y ataques cibernéticos: La protección de los datos de los pacientes y la infraestructura digital es clave para el uso seguro y eficaz de los modelos predictivos y la gestión de demanda.</p> <p>Brecha digital en la población: Limita el acceso equitativo a servicios predictivos y digitales, lo que podría generar desigualdades en el manejo de la demanda.</p> <p>Leyes y normas que impactan la sostenibilidad financiera: Pueden restringir la inversión en nuevas</p>

prepararse para picos de demanda en situaciones de emergencia.	tecnologías y recursos para mejorar la anticipación de demanda.
Infraestructura y equipamiento médico deficiente: Las deficiencias en infraestructura y equipos dificultan la respuesta rápida ante situaciones que sobrepasen la capacidad establecida.	Injerencia de terceros sobre la gestión institucional: Compromete la capacidad de respuesta eficiente y flexible ante situaciones que requieren decisiones rápidas.

A partir de este FODA, se revela que uno de los puntos críticos es la incapacidad para anticipar demanda y garantizar recursos (camas, personal, herramientas) con margen suficiente para picos.

3.1.2 Problemas detectados

Incapacidad para anticipar la demanda de emergencias y hospitalizaciones

Descripción: Aunque EsSalud tiene registros históricos, no siempre se utilizan para modelar la demanda futura de atención en emergencias o internaciones, por lo que no se planifican con suficiente antelación los recursos necesarios (camas, personal, insumos).

Implicancia: Los recursos disponibles (camas, personal) se saturan cuando llega una demanda mayor a lo esperado, sin margen de ajuste rápido.

Consecuencia: Pacientes esperan largas horas para hospitalización; en el Hospital Almenara, por ejemplo, pacientes esperan hasta **96 horas** para atención en emergencias, porque la infraestructura no soporta la demanda institucionalizada. [infobae](#)

Falta de reserva / capacidad extra para responder a picos de demanda

Descripción: No hay suficiente capacidad “ociosa controlada” (camas adicionales, personal en reserva, herramientas de respaldo) que permita gestionar aumentos súbitos de la demanda.

Implicancia: Cuando ocurre un pico (por ejemplo por brotes, aumentos estacionales), se recurre a medidas de emergencia, improvisación o derivaciones.

Consecuencia: Se generan retrasos, saturación de emergencias, transferencia de pacientes, deterioro de calidad de atención.

Insuficiente planificación de recursos humanos ajustada a demanda proyectada

Descripción: La programación del personal médico y de enfermería se hace con base más estática (historias recientes) que con previsiones anticipadas de crecimiento de demanda.

Implicancia: En momentos de alta demanda, no hay personal suficiente para cubrir turnos críticos.

Consecuencia: Sobrecarga del personal, calidad de atención disminuida, tiempos de espera largos.

Planificación de insumos y herramientas basada en consumo histórico sin margen de proyección de demanda oculta

Descripción: La adquisición de insumos tiende a basarse en consumos anteriores, sin incorporar márgenes adicionales para cubrir demanda latente o emergente (“demanda oculta”) estudiada en la literatura para EsSalud (estimaciones de un 20-30 % adicional) [Dialnet](#)

Implicancia: El stock se agota rápido en escenarios de alta demanda.

Consecuencia: Desabastecimientos, suspensión de procedimientos, mala gestión.

Sistemas de información fragmentados que dificultan predicción agregada de demanda

Descripción: Los datos de emergencias, atención externa, hospitalización, camas disponibles, personal, etc., muchas veces están en sistemas distintos que no están integrados o actualizados en tiempo real.

Implicancia: No se puede tener una visión en tiempo real ni proyectiva del uso de recursos ni anticipar cuellos de botella.

Consecuencia: Decisiones reactivas, improvisación, recursos mal distribuidos.

3.2. Priorización de problemas

Problema	Impacto en la misión (anticipar demanda)	Esfuerzo para resolver parcial	Prioridad
Incapacidad para anticipar demanda de emergencias / hospitalizaciones	Muy alto	Alto	Alta
Falta de capacidad de reserva para picos	Alto	Medio-Alto	Alta
Programación de personal no ajustada	Medio-Alto	Medio	Media-Alta
Planificación de insumos con márgenes insuficientes	Medio	Medio	Media
Sistemas de información fragmentados	Medio	Alto	Media

3.3. Filtrado de problemas

Para asegurar que ese enfoque “anticipar demanda” sea factible y tenga base, necesitas filtrar y validar con información:

- Verificar si los hospitales de EsSalud tienen historiales digitales de atenciones (emergencia, hospitalización) con granularidad diaria/horaria.
- Verificar qué tan accesibles son los datos de ocupación de camas, ingresos/egresos diarios, rotación de camas, tiempos de estancia.
- Verificar si existe ya el algoritmo de predicción para emergencias (como el estudio RRI-09-2024) que puede escalar. [IETSI](#)
- Ver si el presupuesto permite contar con una capacidad ociosa mínima (camas de reserva, personal en contingencia).

- Evaluar si hay barreras organizativas, legales o técnicas para integrar sistemas de información entre hospitales, emergencias y atención externa.

Solo los problemas que superen ese filtro (tienen datos, control institucional, margen de acción) pasarán a definición SMART.

3.4. Planteamiento de problemas con criterios SMART

Aquí algunos ejemplos de enunciados SMART focalizados en los problemas prioritarios y recursos:

A) Problema SMART A

“Para diciembre de 2026, EsSalud debe implementar un sistema predictivo (usando series temporales) en al menos 5 hospitales de nivel III, de modo que pueda anticipar la demanda de emergencias con al menos 85 % de exactitud y permitir activar con 48 horas de anticipación la movilización de camas y personal de soporte.”

- *Específico*: predicción de demanda en emergencias.
- *Medible*: 85 % de exactitud, activación con 48 h.
- *Alcanzable*: con modelos ARIMA / SARIMA ya probados en EsSalud. [IETSI](#)
- *Relevante*: permite movilizar recursos antes del pico, evitando saturaciones.
- *Tiempo*: hasta diciembre de 2026.

B) Problema SMART B

“Durante el período 2025-2027, EsSalud incrementará su capacidad de reserva hospitalaria en un 15 % (camas, unidades de soporte, personal en reserva) para responder a picos de demanda no anticipados, con activaciones automáticas basadas en predicción.”

- *Específico*: capacidad de reserva, activación basada en predicción.
- *Medible*: +15 %.
- *Alcanzable*: mediante inversiones y planificación progresiva.
- *Relevante*: reduce saturaciones en picos.
- *Tiempo*: periodo 2025-2027.

C) Problema SMART C

“Para mediados de 2026, la programación del personal médico y de enfermería en hospitales de alta demanda estará alineada con pronósticos semanales generados por sistema predictivo, de modo que al menos el 90 % de turnos críticos estén cubiertos durante semanas de alta demanda proyectada.”

- *Específico*: programación de personal con pronóstico semanal.
- *Medible*: 90 % de cobertura de turnos críticos.
- *Alcanzable*: si el sistema predictivo es preciso.
- *Relevante*: asegura que recursos humanos estén disponibles cuando más se necesitan.
- *Tiempo*: hasta mitad de 2026.

D) Problema SMART D

“Para fines de 2026, los hospitales pilotos integrarán sus sistemas de información (emergencia, hospitalización, camas libres) y generarán reportes automáticos diarios de pronóstico de demanda, permitiendo identificar cada madrugada el riesgo de saturación en las siguientes 48 horas.”

- *Específico*: integración de sistemas, generación de pronósticos.
- *Medible*: reporte diario de riesgo 48 h.
- *Alcanzable*: con arquitectura de datos, ETL / APIs.
- *Relevante*: proporciona visión anticipada para tomar decisiones operativas.
- *Tiempo*: hasta finales de 2026.

NECESIDADES DE INFORMACIÓN Y PREGUNTAS DE NEGOCIO

4.1. Necesidades por nivel de decisión (estratégico/táctico/operativo)

La anticipación de demanda afecta a toda la cadena de decisiones dentro de EsSalud. Aquí se muestran los distintos niveles de decisión y qué tipo de información necesitan.

Nivel	Tipo de decisiones	Necesidades de información / insumos requeridos
Estratégico	<ul style="list-style-type: none"> Definir políticas institucionales para reservas de capacidad hospitalaria Aprobar inversiones en infraestructura, sistemas y tecnología predictiva Establecer criterios de distribución de recursos entre regiones o redes 	<ul style="list-style-type: none"> Proyecciones agregadas nacionales / regionales de demanda hospitalaria (camas, emergencias, internaciones) Modelos de demanda futura (series temporales, escenarios) Análisis de costos-beneficios de mantener reservas Benchmarking con otros países u organismos de salud que usan predicción Indicadores consolidados de ocupación, tasas de uso de camas históricas Sensibilidad ante choques exógenos (epidemias, estacionales)
Táctico	<ul style="list-style-type: none"> Planificación de recursos por hospitales o redes Ajuste de dotaciones de personal, asignación de camas Decidir cuándo activar reservas o medidas de contingencia Priorización de mejoras en ciertos hospitales 	<ul style="list-style-type: none"> Pronósticos semanales/mensuales de ocupación por hospital Niveles de camas libres / ocupadas Indicadores de rotación de camas, días de estancia promedio Datos de ingresos/egresos hospitalarios Datos de personal (turnos, ausencias, capacidad ociosa) Stock y consumo proyectado de insumos (medicamentos, equipos, reactivos) Alertas automáticas de riesgo de saturación
Operativo	<ul style="list-style-type: none"> Ajustes de última hora: cambio de turnos, reubicar pacientes, derivaciones Asignación diaria de camas Gestión del flujo de pacientes en emergencias y hospitalización Monitoreo en tiempo real del uso de recursos 	<ul style="list-style-type: none"> Dashboard en tiempo real con camas disponibles / ocupadas Listas de espera y backlog Alertas inmediatas de congestión Información de variables clave: tasa de llegada de pacientes, frecuencia de ingresos emergentes, urgencias Estado de insumos críticos (disponibilidad actual) Estado del personal asignado: cuántos disponibles en cada turno

4.2. Inventario de preguntas de negocio (por área y prioridad)

A partir de los problemas priorizados (incapacidad de anticipar demanda, falta de reserva, programación de personal, insumos, sistemas de información fragmentados), ahora definimos las principales preguntas de negocio que EsSalud deberá responder para diseñar soluciones.

Las preguntas están agrupadas por área relevante y ordenadas de mayor a menor prioridad dentro de cada área:

Área / dominio	Preguntas de negocio (prioritarias)
Demanda hospitalaria camas /	<ol style="list-style-type: none"> 1. ¿Cuántas camas se van a necesitar diariamente, por hospital / red / región, durante las próximas 1-4 semanas, con un intervalo de confiabilidad (por ejemplo, percentil 90)? 2. ¿Cuál es la tasa de crecimiento esperado de demanda de internaciones por especialidad (medicina interna, cirugía, UCI) en los próximos meses? 3. ¿Cuál es el patrón estacional o de brotes que se ha dado históricamente en emergencias / ingresos hospitalarios? 4. ¿Cuántas camas de reserva deberían mantenerse para cubrir picos de demanda (porcentaje ideal)?
Personal de salud / recursos humanos	<ol style="list-style-type: none"> 5. ¿Cuál debe ser la dotación diaria óptima de médicos, enfermeras y personal de soporte en cada hospital, dados los pronósticos de demanda? 6. ¿Cuántos profesionales adicionales se requieren para manejar escenarios de alta demanda (reservas de personal)? 7. ¿Qué variabilidad en ausencias (enfermedad, permisos, rotación) se debe considerar en la planificación? 8. ¿Cuántos turnos extras o personal de reemplazo se necesitan en escenarios de pico?
Insumos y herramientas	<ol style="list-style-type: none"> 9. ¿Qué volumen de insumos críticos (medicamentos, consumibles, reactivos, dispositivos) será requerido en el horizonte previsto (meses), bajo demanda anticipada? 10. ¿Cuál debe ser el nivel de stock de seguridad para cubrir picos imprevistos? 11. ¿Cuántas herramientas / equipos de respaldo (ventiladores, bombas, monitores) se deben tener disponibles? 12. ¿Cuáles son los cuellos de botella en la cadena de suministro de insumos con base en demanda prevista?
Sistemas / datos / predicción	<ol style="list-style-type: none"> 13. ¿Qué algoritmo de predicción (modelo estadístico / machine learning) ofrece mejor precisión para la demanda hospitalaria de EsSalud? 14. ¿Qué variables históricas (ingresos, estacionalidad, brotes, clima) son más predictivas? 15. ¿Qué grado de integración de datos (emergencia, hospitalización, camas, insumos) se requiere para alimentar el modelo en tiempo real?

	16. ¿Cuál es el horizonte temporal óptimo de predicción (horas, días, semanas) para decisiones operativas / tácticas?
Gestión estratégica / institucional	17. ¿Cuál es el costo estimado de mantener reservas de capacidad (camas y personal) frente al costo de saturaciones y desbordes? 18. ¿Cuál es el retorno en cobertura y mejora en calidad al anticipar demanda vs. reaccionar? 19. ¿Qué políticas institucionales se deben emitir para garantizar que los hospitales adopten el modelo predictivo? 20. ¿En qué hospitales pilotos debe implementarse primero el sistema predictivo para escalar luego?

4.3. Traducción de preguntas en requisitos analíticos

Una vez tenemos las preguntas de negocio, cada una debe traducirse en requisitos analíticos: es decir, ¿qué datos, métricas, modelos o dashboards necesitas construir para responderlas? Aquí algunos ejemplos:

Pregunta de negocio	Requisito analítico (qué construir)
¿Cuántas camas se necesitarán diariamente en las próximas semanas?	Modelo predictivo (serie temporal) por hospital/región que produce pronóstico diario de demanda de camas, con intervalos de confianza; exportación a dashboard táctico.
¿Cuál debe ser la dotación diaria de personal?	Dashboard que relacione pronóstico de demanda con ratios estándar de personal por paciente (por especialidad/hospital).
¿Qué volumen de insumos críticos se requerirá?	Proyección de consumo de insumos por paciente tipo, multiplicado por el pronóstico de demanda. Tabla de requerimientos mensual.
¿Cuál debe ser el nivel de stock de seguridad?	Cálculo del stock de seguridad: por ejemplo, un porcentaje (X %) adicional sobre demanda esperada más variabilidad estimada.
¿Qué algoritmo predictivo usar?	Evaluación comparativa de modelos (ARIMA, SARIMA, regresión, aprendizaje automático) con validación cruzada y métrica (RMSE, MAE).
¿Qué grado de integración de datos se requiere?	Diseño de arquitectura de datos: fuentes (emergencia, hospitalización, camas, insumos), canal ETL, base de datos integrada con frecuencia de actualización, APIs.

¿Cuál es el horizonte temporal óptimo de predicción?	Pruebas analíticas: comparar precisión de pronósticos a 24 h, 48 h, 7 días, 14 días y seleccionar el horizonte con mejor trade-off entre precisión y utilidad.
¿Cuál es el costo/beneficio de reservas frente a saturaciones?	Modelo financiero: comparar escenarios (“sin reservas anticipadas” vs “con reservas anticipadas”) evaluando costos de personal adicional, camas ociosas, costos evitados por saturación (retrasos, atención deficiente).
¿En qué hospitales se implementará primero?	Análisis cluster o priorización: evaluar hospitales con mayores variaciones de demanda, tasas de saturación histórica y factibilidad técnica, para seleccionar pilotos.

KPI'S Y FICHAS TÉCNICAS DE INDICADORES

5.1. Catálogo de KPI's

KPI	Propósito principal
1. Promedio general anual de resultados	Evaluar la evolución global de los valores promedio (p. ej., glucosa) para medir el impacto de políticas de salud a largo plazo.
4. Variación porcentual anual del promedio de resultados	Identificar tendencias de mejora o empeoramiento interanual y orientar decisiones de planificación sanitaria nacional o regional.
9. Proporción de casos con complicaciones	Medir la gravedad de los casos diagnosticados y la efectividad de las estrategias preventivas a nivel macro.
10. Índice de concentración por grupo etario (HHI)	Detectar desigualdades o concentraciones de riesgo en determinados grupos poblacionales para definir políticas de prevención.
2. Tasa de diagnósticos por grupo etario	Focalizar intervenciones en los grupos más afectados y orientar campañas de detección temprana.

3. Departamento con mayor carga de diagnósticos	Priorizar recursos y acciones sanitarias en las regiones con mayor incidencia.
5. Promedio de resultado por tipo de enfermedad	Evaluar diferencias en el control según el tipo de diabetes y ajustar protocolos clínicos.
8. Departamento con mejor control promedio	Identificar buenas prácticas regionales y replicarlas en otras zonas con peor desempeño.
6. Distribución mensual de diagnósticos	Detectar estacionalidad o picos mensuales para ajustar recursos operativos (personal, campañas, insumos).

5.2. Ficha técnica de cada KPI

A. Promedio general de resultado por año

Campo	Descripción
Nombre del KPI	Promedio general de resultado por año
Definición	Mide la tendencia anual de los valores promedio de resultado
Fórmula	Promedio(resultado de exámenes) agrupado por año
Unidad	Valor promedio por año
Frecuencia	Anual
Fuente de datos	Datos Abiertos EsSalud, INEI
Responsable	Unidad de Estadística Médica

B. Tasa de diagnósticos por grupo etario

Campo	Descripción
Nombre del KPI	Tasa de diagnósticos por grupo etario
Definición	Indica qué grupo etario concentra más diagnósticos.
Fórmula	cantidad de diagnósticos / total de diagnósticos * 100

Unidad	% del total de casos
Frecuencia	Mensual
Fuente de datos	Datos Abiertos EsSalud
Responsable	Coordinación Epidemiológica

C. Departamento con mayor carga de diagnósticos

Campo	Descripción
Nombre del KPI	Departamento con mayor carga de diagnósticos
Definición	Identifica la región más afectada.
Fórmula	Suma(cantidad de diagnósticos) por región
Unidad	N° de diagnósticos
Frecuencia	Mensual
Fuente de datos	Datos Abiertos EsSalud, Ubigeo INEI
Responsable	Gerencias Regionales

D. Variación porcentual anual del promedio de resultados

Campo	Descripción
Nombre del KPI	Variación porcentual anual del promedio de resultados
Definición	Mide si los niveles promedio de la enfermedad mejoran o empeoran cada año.
Fórmula	$(\text{Promedio año actual} - \text{Promedio año anterior}) / \text{Promedio año anterior} * 100$
Unidad	% de variación
Frecuencia	Mensual
Fuente de datos	Datos Abiertos EsSalud
Responsable	Dirección de Prevención

E. Promedio de resultado por tipo de diabetes

Campo	Descripción
Nombre del KPI	Promedio de resultado por tipo de diabetes
Definición	Permite detectar si ciertos tipos presentan peores valores.

Fórmula	Promedio de resultado por enfermedad
Unidad	Valor promedio
Frecuencia	Mensual
Fuente de datos	Datos hospitalarios
Responsable	Coordinación de Control Médico

F. Distribución mensual de diagnósticos

Campo	Descripción
Nombre del KPI	Distribución mensual de diagnósticos
Definición	Detecta estacionalidad o picos en diagnósticos.
Fórmula	Total (cantidad de diagnósticos)
Unidad	Nº de diagnósticos por mes
Frecuencia	Mensual
Fuente de datos	Datos Abiertos EsSalud
Responsable	Unidad de Estadística

G. Promedio ponderado de resultados por cantidad de diagnósticos

Campo	Descripción
Nombre del KPI	Promedio ponderado de resultados por cantidad de diagnósticos
Definición	Integra la gravedad promedio considerando el peso de cada registro.
Fórmula	Identifica la región con mejores valores promedio de control.
Unidad	$\Sigma(\text{promedio resultado} \times \text{cantidad diagnósticos}) / \Sigma(\text{cantidad diagnósticos})$
Frecuencia	Mensual
Fuente de datos	Datos Abiertos EsSalud
Responsable	Dirección de Epidemiología

H. Departamento con mejor control promedio

Campo	Descripción
--------------	--------------------

Nombre del KPI	Departamento con mejor control promedio
Definición	Identifica la región con mejores valores promedio de control.
Fórmula	Mínimo (promedio resultado)
Unidad	Valor más bajo promedio
Frecuencia	Mensual
Fuente de datos	RRHH EsSalud, RENIPRESS
Responsable	Jefatura de Recursos Humanos

I. Proporción de casos con complicaciones

Campo	Descripción
Nombre del KPI	Proporción de casos con complicaciones
Definición	Evalúa la gravedad de los diagnósticos según si hay complicaciones
Fórmula	$(\text{Casos con "complicación"} / \text{Total de casos}) \times 100$
Unidad	% de casos complicados
Frecuencia	Mensual
Fuente de datos	RENIPRESS, Datos Abiertos EsSalud
Responsable	Gerencias de Red

J. Índice de concentración por grupo etario

Campo	Descripción
Nombre del KPI	Índice de concentración por grupo etario
Definición	Mide si los diagnósticos están concentrados en un grupo etario o distribuidos.
Fórmula	$\Sigma((\text{cantidad diagnósticos grupo} / \text{total diagnósticos})^2)$
Unidad	Índice (0 = disperso, 1 = concentrado)
Frecuencia	Mensual
Fuente de datos	Datos Abiertos EsSalud, Ubigeo INEI
Responsable	Dirección Estratégica




5.3. Jerarquía de KPI's y semáforos (SLAs / umbrales)

Jerarquía de KPI's

Nivel	KPI	Propósito principal
Estratégico	1. Promedio general anual de resultados	Evaluar la evolución global de los valores promedio (p. ej., glucosa) para medir el impacto de políticas de salud a largo plazo.
Estratégico	4. Variación porcentual anual del promedio de resultados	Identificar tendencias de mejora o empeoramiento interanual y orientar decisiones de planificación sanitaria nacional o regional.
Estratégico	9. Proporción de casos con complicaciones	Medir la gravedad de los casos diagnosticados y la efectividad de las estrategias preventivas a nivel macro.
Estratégico	10. Índice de concentración por grupo etario (HHI)	Detectar desigualdades o concentraciones de riesgo en determinados grupos poblacionales para definir políticas de prevención.
Táctico	2. Tasa de diagnósticos por grupo etario	Focalizar intervenciones en los grupos más afectados y orientar campañas de detección temprana.
Táctico	3. Departamento con mayor carga de diagnósticos	Priorizar recursos y acciones sanitarias en las regiones con mayor incidencia.
Táctico	5. Promedio de resultado por tipo de enfermedad	Evaluar diferencias en el control según el tipo de diabetes y ajustar protocolos clínicos.
Táctico	8. Departamento con mejor control promedio	Identificar buenas prácticas regionales y replicarlas en otras zonas con peor desempeño.

Operativo	6. Distribución mensual de diagnósticos	Detectar estacionalidad o picos mensuales para ajustar recursos operativos (personal, campañas, insumos).
Operativo	7. Promedio ponderado de resultados	Monitorear el desempeño global de manera continua considerando el peso real de cada registro.

Semáforos de KPI's

KPI	Unidad	Óptimo 	Moderado 	Crítico 	Meta	Interpretación
1. Promedio general anual de resultados	mg/dL (por ejemplo, glucosa)	≤ 150	151 – 180	> 180	≤ 150	Evalúa el control promedio de los pacientes a nivel nacional o regional.
2. Tasa de diagnósticos por grupo etario	% del total de casos	$< 25\%$ por grupo	25% – 40%	$> 40\%$	Distribución equilibrada	Indica concentración de casos en un grupo etario; valores altos implican población de riesgo dominante.
3. Departamento con mayor carga de diagnósticos	Nº de diagnósticos	$< 15\%$ del total nacional	15% – 25%	$> 25\%$	$\leq 15\%$	Mide la concentración geográfica de casos; altos valores indican sobrecarga regional.
4. Variación porcentual anual del promedio de resultados	%	$\leq 0\%$ (mejora)	0 – 5%	$> 5\%$	$\leq 0\%$	Mide la tendencia de control de la enfermedad año a año; positivo indica empeoramiento.

5. Promedio de resultado por tipo de enfermedad	mg/dL	≤ 160	161 – 180	> 180	≤ 160	Compara control promedio entre tipos de diabetes; altos valores implican mayor severidad.
6. Distribución mensual de diagnósticos	Nº de casos	$< 10\%$ de variación mensual	10% – 20%	$> 20\%$	$\leq 10\%$	Detecta estacionalidad o brotes; alta variabilidad sugiere picos de incidencia.
7. Promedio ponderado de resultados	mg/dL	≤ 150	151 – 180	> 180	≤ 150	Refleja el control general considerando el peso de los casos; mide el desempeño global.
8. Departamento con mejor control promedio	mg/dL	≤ 140	141 – 160	> 160	≤ 140	Indica la región con mejores resultados promedio; sirve para identificar buenas prácticas.
9. Proporción de casos con complicaciones	% de casos	$\leq 10\%$	11% – 20%	$> 20\%$	$\leq 10\%$	Evalúa la gravedad de los casos; valores altos alertan sobre falta de control o diagnóstico tardío.
10. Índice de concentración por grupo etario (HHI)	Índice (0–1)	≤ 0.20	0.21 – 0.40	> 0.40	≤ 0.20	Mide la concentración de diagnósticos; valores altos indican concentración en pocos grupos.

FUENTES DE DATOS Y INVENTARIO OLTP

6.1. Inventario de sistemas OLTP (tablas, campos clave, frecuencia) (data que tenemos)

El inventario de fuentes OLTP tiene como propósito identificar y documentar los sistemas transaccionales internos o externos que generan los datos crudos esenciales para alimentar los modelos de análisis y predicción en EsSalud.

Esto incluye conocer qué sistema captura cada tipo de dato (por ejemplo, consultas externas, codificación diagnóstica, infraestructura hospitalaria), qué unidad usuaria lo emplea, qué tipo de datos almacena (maestro o transaccional), la tecnología usada para su gestión, con qué frecuencia se actualiza, y cualquier observación relevante (latencia, fiabilidad, dependencia, etc.).

Este inventario es la base para diseñar las conexiones, las extracciones (ETL/ELT), los controles de calidad de datos y los procesos de integración que permitirán que los modelos predictivos (por ejemplo, de demanda hospitalaria) cuenten con datos consistentes, oportunos y verificables.

Sistema	Área usuaria	Tipo	Tecnología	Frecuencia actualización	Observaciones
Sistema de Planeamiento Estratégico – CEPLAN	Planeamiento Estratégico y Estadística	Maestro / referencia geográfica	Archivo XLS	Anual / Semestral (actualizan planes y ubigeos)	Incluye descripciones y metadatos de cada ubigeo; sirve como insumo referencial para consolidar datos de salud y otras áreas del Estado.
Sistema de Consultas Externas – Datos Abiertos EsSalud	Red hospitalaria / Consulta externa	Transaccional (registros médicos de atenciones y diagnósticos)	Archivos CSV	Diario	Incluye registros de exámenes de laboratorio en consultas externas con diagnóstico de diabetes, hipertensión y obesidad (2020–2024).
Sistema de Codificación Geográfica – Ubigeo (INEI)	Planeamiento / Estadística	Maestro (catálogo)	Archivo CSV	Eventual / según necesidad	Contiene códigos y descripciones de distrito, provincia y departamento; se usa como referencia estándar para enlazar datos de salud con ubicación geográfica oficial.
Sistema de Clasificación Internacional de Enfermedades – CIE10	Áreas médicas / estadística	Maestro	Archivo CSV	Eventual / según necesidad	Contiene el código y descripción de enfermedades según la Clasificación Internacional de Enfermedades (CIE-10).

6.2. Tipos de datos y formatos (CSV, JSON, BD relacional, APIs)

Para los sistemas incluidos en la sección 6.1, los formatos y tipos de datos empleados (según fuentes oficiales) son los siguientes:

- **Archivos XLS / Excel:** usados para sistemas maestros de referencia como CEPLAN y RENIPRESS.
- **Archivos CSV:** usados en los sistemas de consulta externa (datos abiertos) y catálogos maestros como Ubigeo y CIE10. En particular, el portal de Datos Abiertos EsSalud publica series históricas de “Atenciones en consulta externa” en formato CSV. [Datos Abiertos+1](#)
- **Sistemas transaccionales clínicos:** en EsSalud se emplea el sistema ESSI (Servicio de Salud Inteligente) para la historia clínica digital, que almacena datos de atenciones, exámenes, diagnósticos, prescripciones. [Facilita+2World Bank+2](#)

- **Plataforma BI / reporte:** EsSalud utiliza Explora EsSalud, una plataforma de business intelligence para generación de tableros y reportes dinámicos. [Explora EsSalud](#)

6.3. Conectividad y consideraciones de seguridad / accesos

Basado en la información oficial de EsSalud y políticas públicas, estas son las consideraciones relevantes para conectividad y seguridad:

- ❖ El sistema **ESSI** se ha implementado para digitalizar la historia clínica en varios establecimientos de EsSalud, lo que implica que los datos clínicos se transmiten entre módulos internos con protocolos seguros. [World Bank+3Essalud+3Gobierno del Perú+3](#)
- ❖ Los datos abiertos publicados (por ejemplo: exámenes de laboratorio para patologías crónicas) se extraen del sistema ESSI y luego se publican en formatos CSV para transparencia. [Datos Abiertos+1](#)
- ❖ En la implementación del ESSI se busca garantizar confidencialidad, integridad y disponibilidad de los datos clínicos, para que los profesionales puedan acceder a atenciones, diagnósticos y exámenes de laboratorio integrados en un único expediente digital. [Essalud+2World Bank+2](#)
- ❖ Los accesos internos a sistemas clínicos (hospitals, emergencias, consultas) deben estar regulados con permisos y controles de usuario, aunque los detalles específicos no son públicos.
- ❖ Para el consumo de datos por parte de la plataforma BI (Explora) deben existir mecanismos de extracción controlada y segura (réplicas, conexiones autorizadas) para evitar afectar el rendimiento del sistema transaccional.
- ❖ Las publicaciones de datos abiertos requieren anonimización o remoción de datos sensibles para cumplir con normativas de privacidad, lo que se evidencia en los datasets disponibles públicos. [Datos Abiertos](#)

6.4. Volúmenes estimados y frecuencia de actualización

Dado que los sistemas OLTP en la tabla 6.1 son mayormente catálogos maestros o conjuntos de datos abiertos (no sistemas clínicos en tiempo real), los volúmenes y frecuencias son modestos:

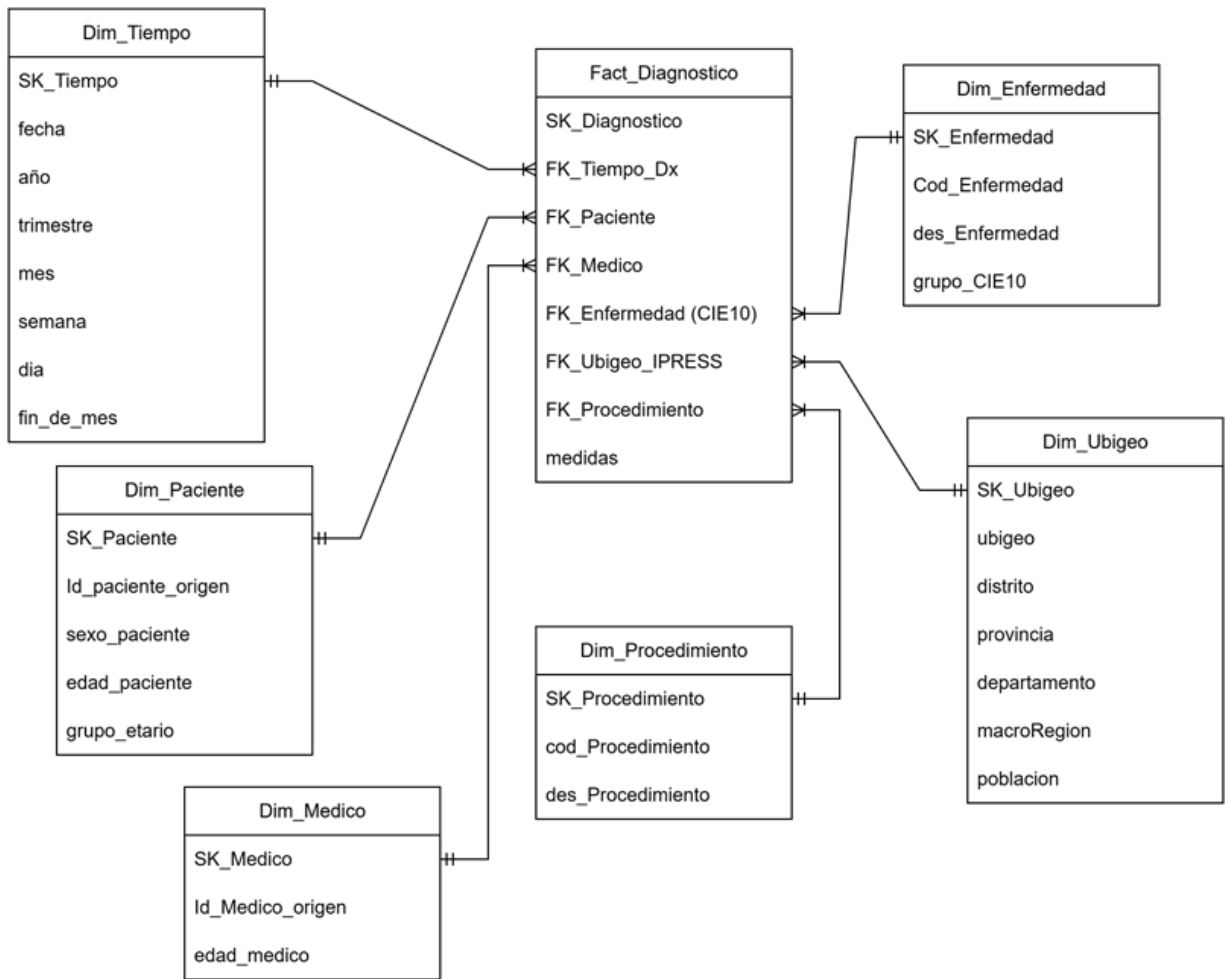
- **Sistema de Planeamiento – CEPLAN:** se actualiza anualmente o semestralmente, con volúmenes pequeños relacionados a catálogos geográficos (ubigeos).
- **Sistema de Consultas Externas – Datos Abiertos EsSalud:** la publicación de “Atenciones en Consulta Externa” contiene miles a decenas de miles de registros históricos; se actualiza cada día con nuevos datos desde ESSI hacia la plataforma de datos abiertos. [Datos Abiertos](#)
- **Sistema de Codificación Geográfica – Ubigeo (INEI):** catálogos geográficos actualizados cuando hay cambios administrativos, con volúmenes relativamente bajos.
- **Sistema CIE10:** catálogo maestro de códigos de enfermedades, que se actualiza cuando la versión oficial cambia.
- **RENIPRESS:** gestión de infraestructura de salud nacional, actualización ocasional cuando se modifican los establecimientos afiliados.

MODELO DE DATOS

7.1. Modelo conceptual (Esquema estrella — diagrama)

El modelo está centrado en la tabla de hechos Fact_Diagnostico, que almacena los eventos de diagnóstico registrados en las consultas externas de EsSalud.

Alrededor de ella se ubican las dimensiones descriptivas, que proporcionan el contexto para analizar los indicadores (KPI) desde distintas perspectivas: tiempo, paciente, médico, enfermedad, IPRESS, ubicación y procedimiento.



7.2. Hechos y dimensiones (lista detallada)

A. Tabla de Hechos:

- Fact_Diagnostico**
 Contiene los eventos cuantificables de diagnóstico de enfermedades crónicas (diabetes, hipertensión, obesidad).
 Cada registro representa un diagnóstico realizado a un paciente en una fecha determinada.
- Claves foráneas (dimensiones):**
 FK_Tiempo_Dx
 FK_Paciente
 FK_Medico
 FK_Enfermedad
 FK_Ubigeo_IPRESS
 FK_Procedimiento

B. Dimensiones:

Dimensión	Descripción / Propósito
-----------	-------------------------

Dim_Tiempo	Permite analizar las métricas por año, trimestre, mes, semana o día.
Dim_Paciente	Describe características demográficas: sexo, edad, grupo etario.
Dim_Medico	Identifica al especialista que realiza el diagnóstico.
Dim_Enfermedad (CIE10)	Clasifica las enfermedades por código y grupo CIE10.
Dim_Ubigeo	Describe la localización geográfica (distrito, provincia, departamento) e incluye la población para calcular tasas por 1000 habitantes.
Dim_Procedimiento	Contiene los procedimientos médicos o exámenes asociados al diagnóstico.

7.3. Atributos, jerarquías y granularidad

A. Atributos principales por dimensión

Dimensión	Atributos relevantes
Dim_Tiempo	fecha, año, trimestre, mes, semana, día, fin_de_mes.
Dim_Paciente	id_paciente_origen, sexo_paciente, edad_paciente, grupo_etario.
Dim_Medico	id_medico_origen, edad_medico.
Dim_Enfermedad	cod_enfermedad, des_enfermedad, grupo_CIE10.
Dim_Ubigeo	ubigeo, distrito, provincia, departamento, macroRegion, poblacion.
Dim_Procedimiento	cod_procedimiento, des_procedimiento.

B. Jerarquías

Dimensión	Jerarquía definida	Uso analítico
------------------	---------------------------	----------------------

Tiempo	Día → Mes → Trimestre → Año	Tendencias temporales y evolución de diagnósticos.
Ubigeo	Distrito → Provincia → Departamento → Macroregión	Análisis geográfico y cálculo de tasas por 1000 hab.
Paciente	Edad → Grupo_Etario	Perfil epidemiológico y segmentación poblacional.
Enfermedad	Código CIE10 → Grupo_CIE10.	Agregación por patología/cluster, comparativos entre grupos de enfermedades..

C. Granulidad

Tabla de hechos: un diagnóstico por paciente, enfermedad y fecha (nivel más bajo).

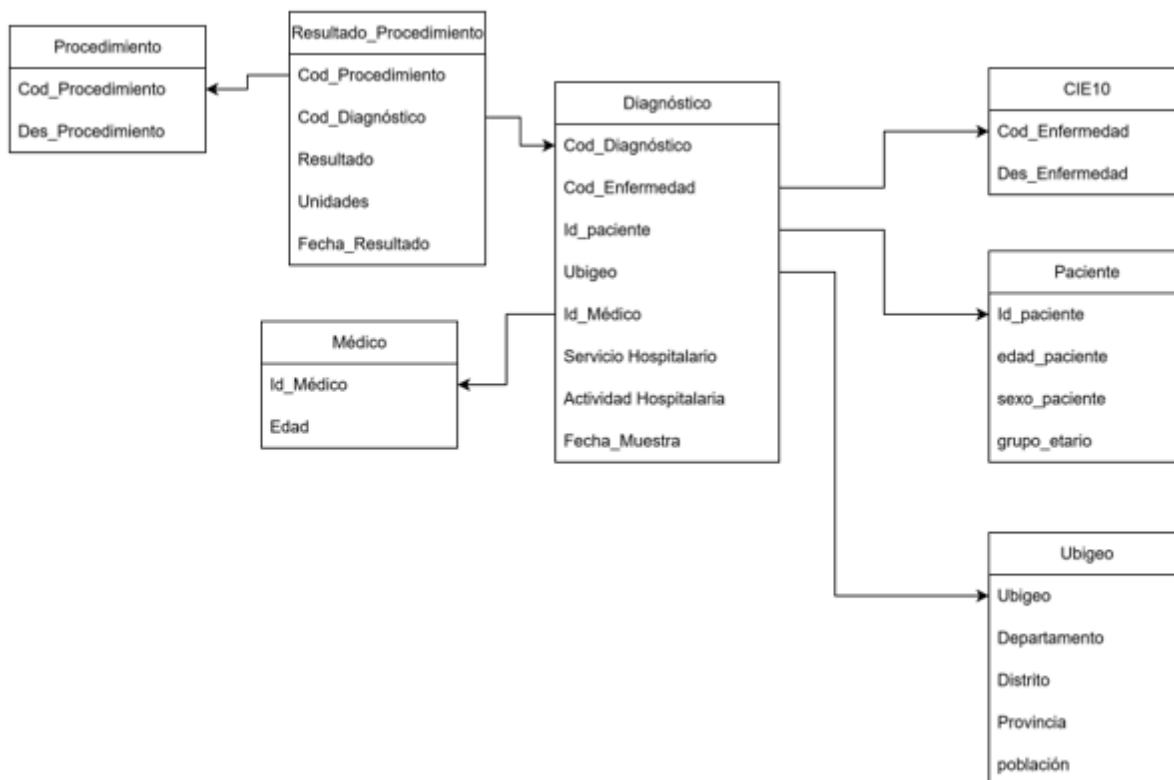
Dim_Tiempo: un registro por día.

Dim_Paciente: un registro por paciente (nivel individual).

Dim_Ubigeo: un registro por distrito.

Dim_Paciente/Médico/Enfermedad/Procedimiento: un registro por entidad de origen

7.4. Modelo lógico / modelo relacional (tablas, claves, índices)



ARQUITECTURA DE SOLUCIÓN ANALÍTICA

8.1. Diagrama de arquitectura end-to-end

La arquitectura propuesta se compone de cuatro capas principales: fuente de datos, ingesta, datawarehouse y consumo. Cada una cumple un rol específico dentro del flujo de procesamiento y análisis de la información proveniente de diversas entidades del sector salud.

8.1.1. Fuente de datos

En esta etapa se concentran las instituciones que generan los datos primarios sobre indicadores de salud y gestión pública. Entre ellas se incluyen:

CEPLAN (Centro Nacional de Planeamiento Estratégico), que provee información estratégica en formato Excel (.xlsx).

EsSalud, que publica datos abiertos en formato CSV desde su plataforma web.

INEI (Instituto Nacional de Estadística e Informática), que proporciona estadísticas sanitarias también en formato CSV.

OMS (Organización Mundial de la Salud), que comparte reportes internacionales descargables en CSV.

SUSALUD (Superintendencia Nacional de Salud), que emite registros administrativos en formato Excel (.xlsx).

Estas fuentes constituyen la base de conocimiento heterogénea a partir de la cual se construye el sistema analítico.

8.1.2. Ingesta

La fase de ingesta se encarga de la recopilación, limpieza y estandarización de los archivos provenientes de las fuentes anteriores.

Los datos se cargan en un ecosistema Big Data soportado por la distribución Hortonworks de Hadoop, que provee la infraestructura base para el almacenamiento distribuido mediante HDFS (Hadoop Distributed File System).

Sobre esta infraestructura se emplean las siguientes herramientas:

Apache Spark, para la transformación masiva y procesamiento paralelo de datos estructurados y semiestructurados.

Apache Hive, que permite realizar consultas SQL sobre los datos almacenados en Hadoop, facilitando la integración con herramientas analíticas posteriores.

De esta forma, los datos en formatos CSV y XLSX se unifican y se almacenan de manera estructurada dentro del ecosistema Hadoop.

8.1.3. Datawarehouse

Una vez procesados y depurados, los datos son almacenados en un repositorio analítico tipo Data Warehouse, implementado con SQLite.

Este motor ligero permite la persistencia de datos consolidados y facilita la conexión directa con herramientas de inteligencia de negocio.

A continuación, Power BI se conecta al almacén de datos para la elaboración de modelos analíticos, dashboards y reportes interactivos, ofreciendo una capa de análisis accesible y visualmente comprensible.

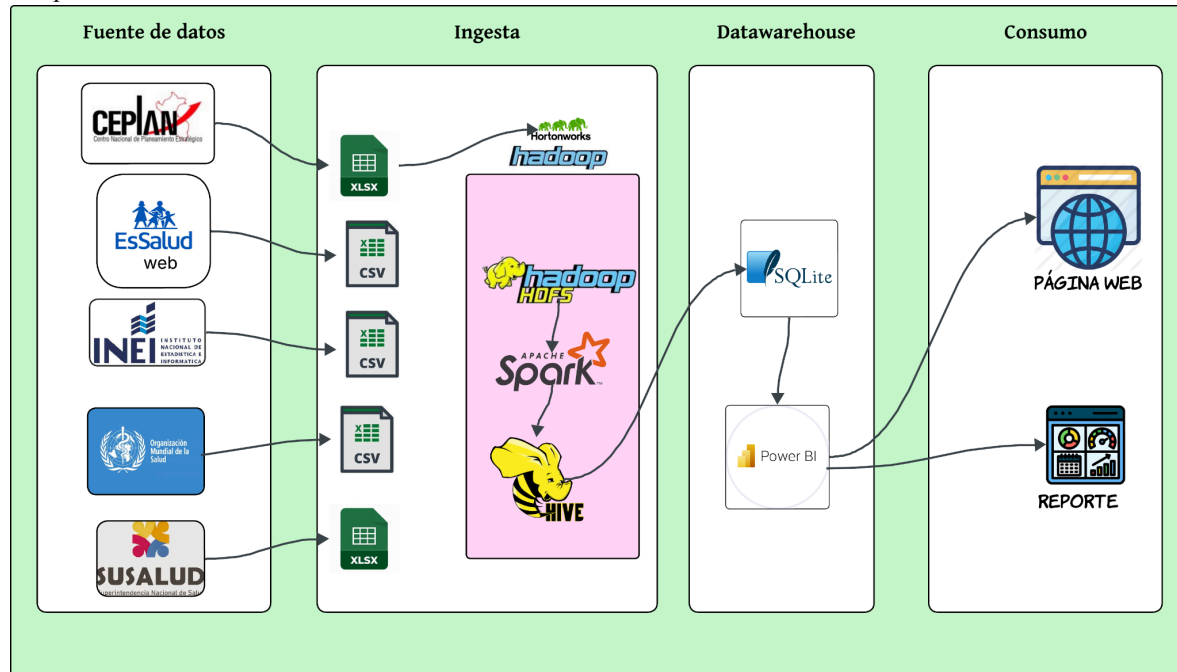
8.1.4. Consumo

La última capa está orientada al consumo de información por parte de los usuarios finales.
A partir de los modelos desarrollados en Power BI y los datos del Data Warehouse, se generan:

Páginas web interactivas, donde se publican indicadores dinámicos.

Reportes ejecutivos y cuadros de mando, que permiten la toma de decisiones basada en evidencia.

De esta manera, la información fluye desde fuentes oficiales hasta herramientas de visualización y consulta en tiempo casi real.



8.2. Componentes: Hortonworks (HDFS, Hive), Spark, Ambari, etc.

Hortonworks Data Platform: La arquitectura de datos propuesta se sustenta sobre un ecosistema Big Data basado en Hortonworks Data Platform (HDP), un entorno de código abierto que integra componentes del ecosistema Apache Hadoop para la gestión, procesamiento y análisis de grandes volúmenes de datos. Este entorno permite la implementación de flujos de ingesta, almacenamiento y procesamiento distribuido, garantizando escalabilidad, disponibilidad y seguridad.

Hadoop Distributed File System: Uno de los pilares de la plataforma es HDFS (Hadoop Distributed File System), el sistema de archivos distribuido diseñado para almacenar grandes conjuntos de datos de manera redundante y tolerante a fallos. HDFS divide los archivos en bloques y los replica en distintos nodos del clúster, lo que asegura la disponibilidad incluso ante fallas de hardware. Su arquitectura maestro-esclavo, compuesta por un NameNode y múltiples DataNodes, permite el acceso concurrente a los datos con alto rendimiento (Shvachko et al., 2010).

Apache Hive: El componente Apache Hive actúa como un motor de almacenamiento y consulta que transforma datos estructurados alojados en HDFS en tablas accesibles mediante un lenguaje similar a SQL, denominado HiveQL. Gracias a su integración con Hadoop, Hive traduce las consultas SQL en trabajos MapReduce o Spark, facilitando el análisis de grandes volúmenes de datos sin requerir programación distribuida. Además, su uso es fundamental para la creación de zonas intermedias (staging y curated) en el proceso de refinamiento de datos (Thusoo et al., 2010).

Apache Spark: Por su parte, Apache Spark es el motor de procesamiento en memoria que reemplaza en muchos casos al modelo MapReduce tradicional debido a su velocidad y flexibilidad. Spark permite el procesamiento batch y streaming, ofreciendo librerías especializadas como Spark SQL, MLlib (para aprendizaje automático), GraphX

(para grafos) y Structured Streaming. En esta arquitectura, Spark se utiliza tanto para la transformación de datos en las zonas staging y curated como para alimentar modelos analíticos o dashboards en tiempo casi real (Zaharia et al., 2016).

Apache Ambari: La administración del clúster se realiza mediante Apache Ambari, herramienta que proporciona una interfaz gráfica y APIs REST para la instalación, configuración, monitoreo y mantenimiento del ecosistema Hadoop. Ambari simplifica tareas complejas como el despliegue de servicios, la gestión de nodos y la supervisión de métricas de rendimiento, garantizando la operatividad continua del entorno distribuido. Su integración con sistemas de autenticación y control de accesos también permite una gestión más segura de los recursos del clúster (Apache Software Foundation, 2023).

De manera complementaria, el ecosistema Hortonworks puede incluir otros servicios como Apache Ranger, encargado de la gestión de políticas de seguridad y auditoría a nivel de datos, y Apache Atlas, utilizado para la gobernanza y trazabilidad de metadatos. Estas herramientas fortalecen el cumplimiento normativo y la administración del ciclo de vida de la información dentro del entorno analítico.

En conjunto, los componentes descritos conforman una plataforma integral que posibilita la gestión eficiente de datos desde su captura hasta su análisis avanzado, alineándose con las mejores prácticas de ingeniería de datos modernas.

8.3. Zonas de datos: raw/staging/curated/analytics

La arquitectura de datos se organiza en diferentes zonas o capas que estructuran el flujo de información desde su origen hasta su consumo analítico. Cada zona cumple una función específica dentro del ciclo de vida del dato, asegurando trazabilidad, control de calidad y optimización de procesos.

Raw: A continuación, los datos pasan a la zona raw, que conserva la información en su estado casi original, pero estructurada dentro del entorno de almacenamiento distribuido, como HDFS. En esta fase, se registran metadatos básicos y se asegura la integridad de los datos, siendo la base para auditorías o reprocesamientos posteriores.

Final: Posteriormente, en la zona staging, los datos son limpiados, normalizados y enriquecidos mediante procesos de transformación (ETL o ELT). Aquí se eliminan duplicados, se corrigen inconsistencias y se realizan uniones entre distintas fuentes. Herramientas como Apache Spark y Hive desempeñan un papel central en esta etapa, ejecutando operaciones distribuidas de integración y validación.

Curated: En la zona curated, los datos transformados se consolidan y almacenan en estructuras optimizadas para el análisis, con modelos normalizados o desnormalizados según los requerimientos del negocio. Es la fuente confiable (single source of truth) para las áreas de análisis y reportes.

Analytics: Finalmente, la zona analytics corresponde al entorno de explotación de datos, donde se alojan los datasets preparados para el consumo por herramientas de inteligencia de negocios, dashboards o modelos de machine learning. Esta capa puede integrarse con soluciones como Power BI, Tableau o plataformas analíticas en la nube, asegurando disponibilidad, rendimiento y consistencia en la entrega de información.

8.4. Patrones de ingesta

El proceso de ingesta de datos se define según la naturaleza de las fuentes y la frecuencia requerida por los casos de uso. En la arquitectura planteada se emplean tres patrones principales: batch, micro-batch y streaming.

El patrón batch es el más utilizado cuando se manejan grandes volúmenes de datos que no requieren actualización inmediata. Este enfoque permite procesar lotes de información a intervalos definidos (por ejemplo, diario o semanal), siendo ideal para informes estadísticos, históricos o de planeamiento estratégico. Hadoop y Spark proporcionan mecanismos eficientes para este tipo de procesamiento distribuido.

El patrón micro-batch combina la estabilidad del procesamiento batch con la necesidad de una mayor frecuencia de actualización. Los datos se ingieren en intervalos cortos, como cada pocos minutos o segundos, lo que permite

mantener sistemas analíticos casi en tiempo real sin la complejidad completa del streaming. Este enfoque es implementado comúnmente mediante Spark Structured Streaming o NiFi.

Finalmente, el patrón streaming se utiliza para el procesamiento continuo de flujos de datos en tiempo real, adecuado para casos de monitoreo, análisis de sensores o alertas. Las herramientas de Apache Kafka y Spark Streaming posibilitan el manejo de eventos de manera inmediata, garantizando baja latencia y escalabilidad.

Estos tres patrones pueden coexistir dentro de la arquitectura, permitiendo adaptarse tanto a fuentes estáticas (como registros médicos históricos) como a flujos dinámicos (como consultas diarias o reportes operativos).

8.5. Seguridad, gobernanza y control de accesos

La seguridad y gobernanza de los datos son ejes fundamentales dentro de la arquitectura. Se implementa un modelo de control basado en roles (Role-Based Access Control, RBAC) que define permisos de acceso según el perfil del usuario (analista, administrador, auditor, entre otros).

En el ecosistema Hortonworks, la herramienta Apache Ranger gestiona las políticas de seguridad de manera centralizada. Permite establecer quién puede leer, escribir o modificar determinados conjuntos de datos, tanto en HDFS como en Hive o Kafka. Además, mantiene registros detallados de auditoría, lo cual es esencial para cumplir con normativas de privacidad y trazabilidad.

Por otro lado, Apache Atlas desempeña un rol clave en la gobernanza de datos, proporcionando un catálogo de metadatos que permite rastrear el linaje del dato desde su origen hasta su uso final. Esto facilita la comprensión del flujo de información y mejora la transparencia en los procesos analíticos.

La seguridad también se refuerza mediante la integración con sistemas de autenticación como Kerberos, que garantiza la identidad de los usuarios y servicios, y la encriptación tanto en tránsito como en reposo. De esta manera, la arquitectura no solo asegura la confidencialidad y disponibilidad de los datos, sino también su integridad y cumplimiento normativo (Apache Software Foundation, 2023).

8.6. Plan de respaldo y recuperación

El plan de respaldo y recuperación busca garantizar la continuidad operativa del sistema y la preservación de los datos ante fallas o pérdidas. Este plan se compone de estrategias de backup, retención y restauración definidas según las políticas institucionales y la criticidad de cada zona de datos.

En el entorno Hadoop, los respaldos pueden implementarse mediante copias incrementales o totales del sistema HDFS utilizando herramientas como DistCp (Distributed Copy), que permite replicar grandes volúmenes de información entre clústeres. Asimismo, Ambari facilita la automatización de respaldos de metadatos, configuraciones y bases de datos asociadas (Apache Software Foundation, 2023).

Las políticas de retención de datos se definen en función del tipo de información: los datos de la zona landing suelen conservarse por períodos cortos (por ejemplo, 7 a 30 días), mientras que los datos de las zonas curated o analytics se retienen durante años por su relevancia histórica y analítica.

Finalmente, el plan de recuperación ante desastres (Disaster Recovery, DR) contempla la existencia de un clúster secundario o una copia replicada en una ubicación alterna. Este esquema permite restaurar rápidamente la operación del sistema ante eventos críticos, minimizando la pérdida de información y el tiempo de inactividad.

DISEÑO DEL PROCESO ETL / ELT

9.1. Diagrama de flujo ETL (Extract - extraer de los csvs cargados y limpieza) → Transform (pasar las fechas de string a datE, agregar el grupo etario y agregar código de cod_institucion en diagnostico) → Load)

Tabla	Descripción	Clave primaria	Relaciones
-------	-------------	----------------	------------

Dim_Enfermedad (df_CIE)	Catálogo CIE10	cod_enfermedad	↔ Diagnóstico
Dim_Paciente (df_Paciente_final)	Pacientes únicos	id_paciente	↔ Diagnóstico
Dim_Medico (df_Medico_final)	Médicos únicos	id_medico	↔ Diagnóstico
Dim_Procedimiento (df_Procedimiento)	Catálogo de exámenes	cod_procedimiento	↔ Resultado_Procedimiento
Dim_IPRESS (df_Ipress)	Establecimientos	cod_institucion	↔ Diagnóstico
Dim_Ubigeo (df_Geodir)	Ubicación geográfica	ubigeo	↔ IPRESS, Diagnóstico
Fact_Diagnostico (df_diagnostico)	Registro principal de atención médica	cod_diagnostico	↔ Todas las dimensiones
Fact_Resultado_Procedimiento (df_Resultado_Procedimiento)	Resultados de laboratorio	(cod_diagnostico, cod_procedimiento)	↔ Diagnóstico y Procedimiento

9.2. Zonas y tablas objetivo (raw → curated)

Zona RAW (Datos Brutos)

Esta zona almacena los datos originales extraídos directamente desde las fuentes (archivos CSV, APIs, bases externas). No se aplican transformaciones ni limpieza significativa, solo se adaptan al formato compatible con el Data Lake (en este caso, formato Parquet).

Estas tablas permiten mantener un histórico íntegro y trazable de la información fuente.

Tablas en zona RAW:

Tabla	Descripción	Ubicación
raw.diagnostico	Contiene diagnósticos médicos registrados, con identificadores de paciente, médico e institución.	/user/hive/warehouse/raw/diagnostico
raw.paciente	Datos de pacientes, incluyendo edad y sexo.	/user/hive/warehouse/raw/paciente

raw.resultado_procedimiento	Resultados de procedimientos médicos realizados.	/user/hive/warehouse/raw/resultado_procedimiento
raw.procedimiento	Catálogo de procedimientos con su descripción.	/user/hive/warehouse/raw/procedimiento
raw.ipress	Información de instituciones de salud (IPRESS).	/user/hive/warehouse/raw/ipress
raw.ubigeo	Tabla de ubicaciones geográficas (departamento, provincia, distrito).	/user/hive/warehouse/raw/ubigeo
raw.CIE	Clasificación de enfermedades según código CIE10.	/user/hive/warehouse/raw/CIE

Zona CURATED (Datos Depurados y Modelados)

En esta zona se almacenan los datos transformados, limpios y enriquecidos, listos para el análisis o para su uso en modelos OLAP o dashboards.

Se agregan campos derivados como grupo etario, se convierten tipos de datos (por ejemplo, fechas), y se integran las distintas fuentes mediante joins lógicos.

Tabla objetivo principal:

Tabla	Descripción	Ubicación
curated.fact_diagnostico	Tabla de hechos que consolida la información de diagnóstico, paciente, procedimiento y localización. Se encuentra particionada por departamento para optimizar consultas analíticas.	/user/hive/warehouse/curated/fact_diagnostico

Campos de la tabla fact_diagnostico:

Campo	Descripción
--------------	--------------------

cod_diagnostico	Identificador del diagnóstico
id_paciente	Código único del paciente
sexo_paciente	Género del paciente
grupo_etario	Grupo de edad calculado a partir de edad_paciente
cod_institucion	Código de la IPRESS donde se realizó la atención
nombre	Nombre de la institución
departamento	Departamento del establecimiento
red	Red de salud correspondiente
cod_enfermedad	Código CIE10 de la enfermedad
des_enfermedad	Descripción de la enfermedad
des_procedimiento	Descripción del procedimiento realizado
resultado	Resultado del procedimiento (positivo, negativo, etc.)
unidades	Unidades del procedimiento
fecha_resultado	Fecha en que se obtuvo el resultado del procedimiento

9.3. Reglas de negocio y transformaciones clave (normalizaciones, deduplicación)

Durante el proceso ETL se aplicaron una serie de reglas de negocio y transformaciones de datos para garantizar la calidad, consistencia y usabilidad de la información dentro del Data Lake y posterior modelo OLAP.

A. Normalización de estructuras

- Se homogenizaron los nombres de columnas y tipos de datos entre las distintas fuentes (df_diagnostico, df_paciente, df_resultado_procedimiento, etc.) para facilitar los joins.
- Se convirtieron las variables de texto con mayúsculas inconsistentes a formato upper() (por ejemplo, nombres de departamentos o redes de salud).

- Las fechas se transformaron del tipo string → date, utilizando formatos estándar (yyyy-MM-dd).

B. Enriquecimiento de datos

- Se añadió la columna grupo_etario, derivada del campo edad_paciente, aplicando las siguientes reglas:

Rango de edad	Grupo etario asignado
0 – 11 años	Niño
12 – 17 años	Adolescente
18 – 29 años	Joven
30 – 59 años	Adulto
60+ años	Adulto Mayor

- Se incorporó el código de institución (cod_institucion) en la tabla de diagnóstico, tomando como referencia la tabla df_ipress.
- Se unificó el catálogo de enfermedades (CIE10) mediante la tabla df_CIE, añadiendo la descripción de enfermedad (des_enfermedad).

C. Depuración y control de calidad

- Se eliminaron registros con campos críticos nulos (dropna), especialmente en columnas como resultado, cod_diagnostico o id_paciente.
- Se eliminaron duplicados mediante:


```
df_fact = df_fact.dropDuplicates(["cod_diagnostico", "id_paciente", "fecha_resultado"])
```
- Se descartaron fechas fuera de rango lógico (por ejemplo, años futuros o menores a 1900).

D. Validación de integridad referencial

- Todos los códigos (id_paciente, cod_diagnostico, cod_institucion, cod_enfermedad) fueron validados para existir en su tabla maestra correspondiente.
- Los joins fueron realizados en modo left join para preservar la información base de diagnóstico, incluso si faltaban datos secundarios.

E. Transformaciones finales

- Selección y reordenamiento de columnas relevantes para el modelo de análisis.
- Escritura del resultado en la zona Curated, en formato Parquet, particionado por departamento:

```
df_fact.write.mode("overwrite").partitionBy("departamento").parquet("/user/hive/warehouse/curated/fact_diagnostico")
```

9.4. Estrategia de particionado y formato (Parquet, compresión, particiones)

Con el objetivo de optimizar el rendimiento de lectura, almacenamiento y procesamiento analítico de los datos en la zona Curated, se definió una estrategia de almacenamiento eficiente basada en el formato Parquet y en la partición por campos de segmentación geográfica.

A. Formato de almacenamiento: Parquet

- Se seleccionó el formato Parquet por sus ventajas frente a los formatos planos (CSV, JSON):
 - Compresión columnar eficiente, reduciendo el espacio en disco.
 - Lectura selectiva de columnas (column pruning).
 - Compatibilidad nativa con motores como Hive, Spark SQL y Presto.
- El formato Parquet permite mantener esquemas estructurados y metadatos útiles para análisis OLAP.

Ejemplo de escritura:

```
df_fact.write.mode("overwrite").partitionBy("departamento").parquet("/user/hive/warehouse/curated/fact_diagnostico")
```

B. Estrategia de particionado

- Se definió la partición primaria por la columna departamento, dado que esta variable:
 - Representa una dimensión geográfica natural en los análisis de salud pública.
 - Facilita consultas filtradas por región (lectura paralela en Hadoop/Spark).
 - Mejora el rendimiento al evitar el escaneo completo del dataset.

Estructura resultante:

```
/user/hive/warehouse/curated/fact_diagnostico/  
├── departamento=LIMA/  
├── departamento=AREQUIPA/  
├── departamento=PIURA/  
└── ...
```

- En caso de ampliación futura, se considera un doble particionado (por ejemplo, departamento + año), lo cual permitirá optimizar análisis temporales o regionales.

C. Compresión

- Se aplicó compresión Snappy, recomendada para Parquet debido a su:
 - Balance entre velocidad y tasa de compresión.

- Descompresión rápida, ideal para consultas interactivas.
- Compatibilidad con los principales frameworks Hadoop.

Configuración en Spark (opcional):

```
spark.conf.set("spark.sql.parquet.compression.codec", "snappy")
```

D. Ventajas esperadas

Aspecto	Beneficio
Lectura selectiva	Solo se cargan columnas requeridas por la consulta.
Filtrado por partición	Los queries sobre un departamento leen únicamente su carpeta.
Almacenamiento optimizado	Reducción significativa del tamaño en disco.

9.5. Manejo de errores y reintentos (dead-letter queue)

En el proceso ETL implementado, se estableció una estrategia de control de calidad y manejo de errores con el fin de garantizar la integridad, trazabilidad y confiabilidad de los datos cargados desde la zona Raw hacia la zona Curated.

1. Detección de errores en extracción (Extract)

Durante la carga inicial desde los archivos CSV hacia Spark, se validan posibles inconsistencias de origen:

- **Archivos corruptos o con delimitadores inválidos.**
→ Se aplican bloques try-except para capturar errores de lectura.
- **Columnas faltantes o con nombres distintos al esperado.**
→ Se emplea validación del esquema (df.schema).
- **Filas vacías o duplicadas.**
→ Se filtran antes de ingresar a la zona Raw.

Ejemplo en PySpark:

```
try:
    df_diagnostico = spark.read.csv("/data/diagnostico.csv", header=True, inferSchema=True)
except Exception as e:
    # Registro de error
    with open("/user/hive/warehouse/logs/etl_errors.log", "a") as log:
```

```
log.write(f'Error al cargar diagnostico.csv: {str(e)}\n')
```

2. Manejo de errores en transformación (Transform)

Durante las transformaciones en Spark:

- **Errores de tipo de datos (fecha, numérico):**
Se convierten mediante `to_date()` y `cast()`, y los registros que no cumplan las reglas se envían a una zona de cuarentena.
- **Registros con datos incompletos en campos críticos** (por ejemplo, `cod_diagnostico`, `fecha_resultado`, `resultado`):
→ Se excluyen mediante `dropna()` y se almacenan en una tabla de errores o dead-letter table.

Ejemplo:

```
df_errores = df_fact.filter(df_fact["resultado"].isNull())
df_errores.write.mode("append").parquet("/user/hive/warehouse/errors/fact_diagnostico_error")
df_fact = df_fact.dropna(subset=["resultado"])
```

3. Dead-Letter Queue (DLQ)

Se implementó una estructura de almacenamiento dedicada a los registros rechazados, denominada Dead-Letter Queue (DLQ).

Su propósito es almacenar los registros con errores críticos sin interrumpir el flujo principal de carga.

Estructura:

```
/user/hive/warehouse/errors/
├── diagnostico_error/
├── paciente_error/
└── procedimiento_error/
```

Cada tabla contiene las siguientes columnas:

- `registro_original`: JSON completo del registro rechazado.
- `error_tipo`: Descripción del error detectado.
- `fecha_registro`: Fecha y hora en que ocurrió el error.

4. Política de reintentos

Los registros en la Dead-Letter Queue se pueden reprocesar manual o automáticamente:

- **Automático:** mediante un job programado en Spark/Hive que intenta volver a cargar los registros después de correcciones.
- **Manual:** revisando el log `/user/hive/warehouse/logs/etl_errors.log` y actualizando los CSV de origen.

Ejemplo de reintento automático:


```
df_retry = spark.read.parquet("/user/hive/warehouse/errors/fact_diagnostico_error")
df_retry_clean = df_retry.filter(df_retry["resultado"].isNotNull())
df_retry_clean.write.mode("append").parquet("/user/hive/warehouse/curated/fact_diagnostico")
```

5. Beneficios del enfoque DLQ

Aspecto	Beneficio
Trazabilidad	Cada error queda registrado con su causa.
Disponibilidad	El flujo ETL no se interrumpe ante errores individuales.
Calidad de datos	Los registros defectuosos son aislados y corregidos posteriormente.
Reprocesamiento controlado	Los registros válidos pueden reincorporarse sin repetir toda la carga.

IMPLEMENTACIÓN TÉCNICA EN HORTONWORKS

10.1. Ingesta inicial a HDFS

En esta sección se describe el procedimiento realizado para la ingesta inicial de archivos al sistema distribuido de archivos de Hadoop (HDFS) dentro del entorno Hortonworks.

El objetivo es automatizar la transferencia de archivos locales (en formatos .csv y .xlsx) hacia la zona /data/raw/ del HDFS, garantizando trazabilidad y reproducibilidad del proceso.

10.1.1. Configuración de conexión SSH entre host y Hortonworks

Para establecer una comunicación segura entre la máquina anfitriona (Windows) y la máquina virtual (Hortonworks Sandbox), se configuró un acceso mediante llaves SSH.

PASO 1: Creación de la carpeta para almacenar la llave

Código:

```
mkdir $env:USERPROFILE\.ssh
```

Ejecución:

```
PS C:\Users\Lenovo> mkdir $env:USERPROFILE\.ssh

Directorio: C:\Users\Lenovo

Mode                LastWriteTime         Length Name
----                -
d-----         12/10/2025   11:46         .ssh
```

Figura X. Ejecución. Creación de la carpeta para almacenar la llave de acceso

PASO 2: Generación de las llaves RSA de 4096 bits

Código:

```
ssh-keygen -t rsa -b 4096 -f $env:USERPROFILE\.ssh\horton_id_rsa
```

Ejecución:

```
PS C:\Users\Lenovo> ssh-keygen -t rsa -b 4096 -f $env:USERPROFILE\.ssh\horton_id_rsa
Generating public/private rsa key pair.
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\Lenovo\.ssh\horton_id_rsa
Your public key has been saved in C:\Users\Lenovo\.ssh\horton_id_rsa.pub
The key fingerprint is:
SHA256:bMi4xmNysAIu/yQbEQIrnC1/ocSfoV4go18LX0kaXVg lenovo@DESKTOP-29DDJCL
The key's randomart image is:
+---[RSA 4096]-----+
|  .  .+.E          |
| o.+  +o.         |
| =*o=.oo o        |
| =.*+==++         |
| +.++==+ S        |
| .+.o+ .          |
| =.o +            |
| * . .            |
| .                |
+---[SHA256]-----+
```

Figura X. Ejecución. Generación de las llaves RSA

PASO 3: Verificación de los archivos generados

Código:

```
ls $env:USERPROFILE\.ssh
```

Ejecución:

```
PS C:\Users\Lenovo> ls $env:USERPROFILE\.ssh

Directorio: C:\Users\Lenovo\.ssh

Mode                LastWriteTime         Length Name
----                -
-a----             12/10/2025   11:47           3389 horton_id_rsa
-a----             12/10/2025   11:47             749 horton_id_rsa.pub
```

Figura X. Ejecución. Verificación de los archivos generados

Donde:

- horton_id_rsa corresponde a la llave privada, almacenada localmente.
- horton_id_rsa.pub es la llave pública, que se copia al servidor remoto.

PASO 4: Obtención de la dirección IP de la máquina virtual

Código:

```
ip a
```

Ejecución:

```
[root@sandbox-hdp ~]# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
5: eth0@if6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
    link/ether 02:42:ac:12:00:02 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 172.18.0.2/16 brd 172.18.255.255 scope global eth0
        valid_lft forever preferred_lft forever
```

Figura X. Ejecución. Obtención de la dirección IP de la máquina virtual

PASO 5: Conexión al entorno virtual Hortonworks desde PowerShell

Código:

```
ssh -p 2222 root@127.0.0.1
```

Ejecución:

```
PS C:\Users\Lenovo> ssh -p 2222 root@127.0.0.1
The authenticity of host '[127.0.0.1]:2222 ([127.0.0.1]:2222)' can't be established.
ED25519 key fingerprint is SHA256:7C3ELG2dUbGt7trSrx8YYsXHZHRprMe+UC0eIlkxTb0.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '[127.0.0.1]:2222' (ED25519) to the list of known hosts.
root@127.0.0.1's password:
Last login: Sun Oct 12 16:37:51 2025 from 172.18.0.2
[root@sandbox-hdp ~]#
```

Figura X. Ejecución. Conexión al entorno virtual Hortonworks desde PowerShell

PASO 6: Registro de la llave pública en la VM

Código:

```
mkdir -p ~/.ssh
cat /root/horton_id_rsa.pub >> ~/.ssh/authorized_keys
chmod 700 ~/.ssh
chmod 600 ~/.ssh/authorized_keys
```

Ejecución:

```
PS C:\Users\Lenovo> ssh -p 2222 root@127.0.0.1
root@127.0.0.1's password:
Last login: Sun Oct 12 17:22:32 2025 from 172.18.0.3
[root@sandbox-hdp ~]# mkdir -p ~/.ssh
[root@sandbox-hdp ~]# cat /root/horton_id_rsa.pub >> ~/.ssh/authorized_keys
[root@sandbox-hdp ~]# chmod 700 ~/.ssh
[root@sandbox-hdp ~]# chmod 600 ~/.ssh/authorized_keys
[root@sandbox-hdp ~]# exit
logout
Connection to 127.0.0.1 closed.
PS C:\Users\Lenovo> ssh -p 2222 -i C:\Users\Lenovo\.ssh\horton_id_rsa root@127.0.0.1
Last login: Sun Oct 12 17:54:27 2025 from 172.18.0.3
[root@sandbox-hdp ~]# exit
logout
Connection to 127.0.0.1 closed.
```

Figura X. Ejecución. Registro de la llave pública en la VM

10.1.2. Automatización de carga con script Python

Para optimizar la carga de archivos al HDFS, se desarrolló un script en Python denominado `subir_a_hdfs.py`. Este script automatiza la conversión de archivos Excel a CSV, la transferencia a la máquina virtual mediante scp, y finalmente la carga en HDFS.

Código:

- Importación de librerías

```
import os
import re
import unicodedata
import subprocess
import pandas as pd
```

- Configuración de parámetros

```
KEY_PATH = r"C:\Users\Lenovo\.ssh\horton_id_rsa"
LOCAL_DIR = r"C:\Inteligencia_Negocios\Archivos\Descargados"
REMOTE_DIR = "/root/hdfs_upload"
HDFS_DIR = "/data/raw/"
SSH_CONN = "root@127.0.0.1"
SSH_PORT = "2222"
```

10.1.3. Conversión de archivos Excel a CSV

El script incluye una función que normaliza nombres de archivos y convierte las hojas de Excel en archivos CSV independientes, preservando las tildes y caracteres especiales.

```
def limpiar_nombre(nombre):
    nfkd = unicodedata.normalize("NFKD", nombre)
    sin_tildes = "".join([c for c in nfkd if not unicodedata.combining(c)])
    sin_espacios = sin_tildes.replace(" ", "_")
    limpio = re.sub(r"[^A-Za-z0-9._-]", "", sin_espacios)
    return limpio
```

Cada archivo Excel (.xls o .xlsx) es procesado hoja por hoja y exportado a formato CSV:

```
print("📄 Convirtiendo archivos Excel a CSV...")
converted_files = []
for f in files:
    path = os.path.join(LOCAL_DIR, f)
    if f.endswith(("xls", "xlsx")):
        print(f" -> {f}")
        xls = pd.ExcelFile(path)
        for sheet in xls.sheet_names:
            df = pd.read_excel(xls, sheet_name=sheet)
            clean_sheet = limpiar_nombre(sheet)
            new_name = limpiar_nombre(f"{os.path.splitext(f)[0]}_{clean_sheet}.csv")
            new_path = os.path.join(LOCAL_DIR, new_name)
            df.to_csv(new_path, index=False)
            converted_files.append(new_name)
            print(f"   -> Hoja '{sheet}' -> {new_name}")
    else:
        converted_files.append(f)
```

10.1.4. Subida automática a HDFS y verificación

Una vez convertidos los archivos, el script automatiza su transferencia y posterior carga al HDFS.

- Copia de archivos a la máquina virtual

```
print("\n📁 Copiando archivos a la VM (sandbox)...")
subprocess.run(
    ["ssh", "-p", SSH_PORT, "-i", KEY_PATH, SSH_CONN, f'mkdir -p {REMOTE_DIR}'],
    check=True,
)

for f in converted_files:
    local_path = os.path.join(LOCAL_DIR, f)
    print(f"→ Subiendo {f}")
    subprocess.run(
        [
            "scp",
            "-p",
            SSH_PORT,
            "-i",
            KEY_PATH,
            local_path,
            f'{SSH_CONN}:{REMOTE_DIR}/{f}',
        ],
```

```
    check=True,
)
```

- Creación de carpeta en HDFS y carga final}

```
print("\n📁 Subiendo archivos al HDFS...")
cmd_create_dir = f"hdfs dfs -mkdir -p {HDFS_DIR}"
cmd_put = f'for f in {REMOTE_DIR}/*; do hdfs dfs -put -f "$f" {HDFS_DIR}; done'

subprocess.run(
    ["ssh", "-p", SSH_PORT, "-i", KEY_PATH, SSH_CONN, cmd_create_dir], check=True
)
subprocess.run(["ssh", "-p", SSH_PORT, "-i", KEY_PATH, SSH_CONN, cmd_put], check=True)

print("\n✅ ¡Archivos subidos correctamente a HDFS en /data/raw/!")
```

- Verificación del resultado

Se comprobó la correcta carga de los archivos en la ruta /data/raw/ mediante el siguiente comando:
hdfs dfs -ls /data/raw/

Con la siguiente respuesta

```
[root@sandbox-hdp ~]# hdfs dfs -ls /data/raw/
Found 7 items
-rw-r--r--  1 root hdfs      913690 2025-10-12 21:41 /data/raw/CIE10_2021.csv
-rw-r--r--  1 root hdfs  241790497 2025-10-12 21:41 /data/raw/DF_ExLab_CExt_Diabetes.csv
-rw-r--r--  1 root hdfs  49791254 2025-10-12 21:41 /data/raw/DF_ExLab_CExt_EnfermedadRenal.csv
-rw-r--r--  1 root hdfs  46893990 2025-10-12 21:42 /data/raw/DF_ExLab_CExt_Hiperlipidemia.csv
-rw-r--r--  1 root hdfs    839681 2025-10-12 21:41 /data/raw/Datos_planeamiento_estrategico_Distrital.csv
-rw-r--r--  1 root hdfs    120413 2025-10-12 21:42 /data/raw/geodir_ubigeo_inei_ubigeo_inei.csv
-rw-r--r--  1 root hdfs   11945413 2025-10-12 21:42 /data/raw/ipress_Listado_de_Establecimientos.csv
```

Que indica que se subieron exitosamente los datos a la carpeta /data/raw/

TRANSFORMACIONES EN SPARK

11.1. Pipeline de limpieza y enriquecimiento (detalles de transformaciones)

11.1.1 Carga de dataframes

- CIE10_2021.csv

```
%pyspark

df_CIE = spark.read.option("header", "true").csv("/data/raw/CIE10_2021.csv")
df_CIE.printSchema()
df_CIE.show(5)

root
 |-- CODIGO: string (nullable = true)
 |-- DESCRIPCION: string (nullable = true)
+-----+-----+
|CODIGO|DESCRIPCION|
+-----+-----+
| Y21.4|Ahogamiento y sum...|
| Y21.5|Ahogamiento y sum...|
| Y21.6|Ahogamiento y sum...|
| Y21.7|Ahogamiento y sum...|
| Y21.8|Ahogamiento y sum...|
+-----+-----+
only showing top 5 rows
```

- DF_ExLab_CExt_Diabetes.csv

```
%pyspark
df_Diabetes = spark.read.option("header", "true").csv("/data/raw/DF_ExLab_CExt_Diabetes.csv")
df_Diabetes.printSchema()
df_Diabetes.show(5)

|-- FECHA_CORTE;DEPARTAMENTO;PROVINCIA;DISTRITO;UBIGEO;RED;IPRESS;ID_PACIENTE;EDAD_PACIENTE;SEXO_PACIENTE;EDAD_MEDICO;ID_MEDICO;COD_DIAG;DIAGNOSTICO;AREA_HOSPITALARIA;SERVICIO_HOSPITALARIO;ACTIV
IDAD_HOSPITALARIA;FECHA_MUESTRA;FEC_RESULTADO_1;PROCEDIMIENTO_1;RESULTADO_1;UNIDADES_1;FEC_RESULTADO_2;PROCEDIMIENTO_2;RESULTADO_2;UNIDADES_2|
-----+-----
|
|
20240531;UCAVALI;...|
|
20240531;UCAVALI;...|
|
20240531;TACHA;TA...|
|
20240531;TACHA;TA...|
|
20240531;SAN MART...|
+-----+-----
only showing top 5 rows

Took 6 sec. Last updated by anonymous at October 12 2025, 5:00:25 PM.
```

- DF_ExLab_CExt_EnfermedadRenal.csv

```
%pyspark
df_Enf_Renal = spark.read.option("header", "true").csv("/data/raw/DF_ExLab_CExt_EnfermedadRenal.csv")
df_Enf_Renal.printSchema()
df_Enf_Renal.show(5)

root
|-- FECHA_CORTE;DEPARTAMENTO;PROVINCIA;DISTRITO;UBIGEO;RED;IPRESS;ID_PACIENTE;EDAD_PACIENTE;SEXO_PACIENTE;EDAD_MEDICO;ID_MEDICO;COD_DIAG;DIAGNOSTICO;AREA_HOSPITALARIA;SERVICIO_HOSPITALARIO;A
CTIVIDAD_HOSPITALARIA;FECHA_MUESTRA;FEC_RESULTADO_1;PROCEDIMIENTO_1;RESULTADO_1;UNIDADES_1;FEC_RESULTADO_2;PROCEDIMIENTO_2;RESULTADO_2;UNIDADES_2| string (nullable = true)
-----+-----
|FECHA_CORTE;DEPARTAMENTO;PROVINCIA;DISTRITO;UBIGEO;RED;IPRESS;ID_PACIENTE;EDAD_PACIENTE;SEXO_PACIENTE;EDAD_MEDICO;ID_MEDICO;COD_DIAG;DIAGNOSTICO;AREA_HOSPITALARIA;SERVICIO_HOSPITALARIO;ACTIV
IDAD_HOSPITALARIA;FECHA_MUESTRA;FEC_RESULTADO_1;PROCEDIMIENTO_1;RESULTADO_1;UNIDADES_1;FEC_RESULTADO_2;PROCEDIMIENTO_2;RESULTADO_2;UNIDADES_2|
+-----+-----
|
|
20240531;CALLAO;C...|
|
20240531;HUANUCO;...|
|
20240531;HUANUCO;...|
|
20240531;HUANUCO;...|
+-----+-----
Took 1 sec. Last updated by anonymous at October 12 2025, 5:01:29 PM.
```

- DF_ExLab_CExt_Hiperlipidemia.csv

```
%pyspark
df_Enf_Renal = spark.read.option("header", "true").csv("/data/raw/DF_ExLab_CExt_Hiperlipidemia.csv")
df_Enf_Renal.printSchema()
df_Enf_Renal.show(5)

root
|-- FECHA_CORTE;DEPARTAMENTO;PROVINCIA;DISTRITO;UBIGEO;RED;IPRESS;ID_PACIENTE;EDAD_PACIENTE;SEXO_PACIENTE;EDAD_MEDICO;ID_MEDICO;COD_DIAG;DIAGNOSTICO;AREA_HOSPITALARIA;SERVICIO_HOSPITALARIO;A
CTIVIDAD_HOSPITALARIA;FECHA_MUESTRA;FEC_RESULTADO_1;PROCEDIMIENTO_1;RESULTADO_1;UNIDADES_1;FEC_RESULTADO_2;PROCEDIMIENTO_2;RESULTADO_2;UNIDADES_2| string (nullable = true)
-----+-----
|FECHA_CORTE;DEPARTAMENTO;PROVINCIA;DISTRITO;UBIGEO;RED;IPRESS;ID_PACIENTE;EDAD_PACIENTE;SEXO_PACIENTE;EDAD_MEDICO;ID_MEDICO;COD_DIAG;DIAGNOSTICO;AREA_HOSPITALARIA;SERVICIO_HOSPITALARIO;ACTIV
IDAD_HOSPITALARIA;FECHA_MUESTRA;FEC_RESULTADO_1;PROCEDIMIENTO_1;RESULTADO_1;UNIDADES_1;FEC_RESULTADO_2;PROCEDIMIENTO_2;RESULTADO_2;UNIDADES_2|
+-----+-----
|
|
20240602;LIMA;LIM...|
|
20240602;LA LIBER...|
|
20240602;LIMA;HUA...|
|
20240602;ICA;CHIN...|
+-----+-----
```

- Datos_planeamiento_estrategico_Distrital.csv

```
%pyspark
df_Plan_estrategico = spark.read.option("header", "true").csv("/data/raw/Datos_planeamiento_estrategico_Distrital.csv")
df_Plan_estrategico.printSchema()
df_Plan_estrategico.show(5)

null|          64|          540|          22540.0|          -|          8942|          162076.0|          7.24830995198497|126022|
123503.0|
7110|
15.7373603722115|
-|          57.0|          40.3688072729351|          18|          22|          -|          -|          291|          31.8
344146399195|
615|          2690762|          3324781.88777294|          3171351.39|          2947.76052135191|          5.8648998667138|          821.040898295549|          584262.
1964754|          3768.8014
|          Región|          010000|          AMAZONAS|          406087|          417365.0|          472993.0|          8043.0|
7743.0|          7826|          39249.13|12.0510441887502|          Chachapoyas|          2338.0|-6.229444444444445|-77.8727777777777|          -|
null|          7|          84|          3174.0|          Ecuador|          8043.0|
5113.0|
184|          3.28571428571429|          365|          7.13866614512028|          5600|
17.4673888030566|          34.2307430117196|          2|          51.9|          0.868359863758087|          35|
0.6628952|          6.0|          1|          2|          26|          0.41773584000102|          34528.489|
32.95|          155851.1935|          7.3|          1561.3788374458|          5481.20884996184|
124320|          142701.360234441|          3919.83001251604|          1561.3788374458|          5481.20884996184|

Took 1 sec. Last updated by anonymous at October 12 2025, 5:04:06 PM.
```

- geodir_ubigeo_inei_ubigeo_inei.csv

```
%pyspark

df_Geodir = spark.read.option("header", "true").csv("/data/raw/geodir_ubigeo_inei_ubigeo_inei.csv")
df_Geodir.printSchema()
df_Geodir.show(5)
```

```
root
 |-- Ubigeo: string (nullable = true)
 |-- Distrito: string (nullable = true)
 |-- Provincia: string (nullable = true)
 |-- Departamento: string (nullable = true)
 |-- Poblacion: string (nullable = true)
 |-- Superficie: string (nullable = true)
 |-- Y: string (nullable = true)
 |-- X: string (nullable = true)
+-----+-----+-----+-----+-----+-----+-----+
|Ubigeo|  Distrito| Provincia|Departamento|Poblacion|Superficie|      Y|      X|
+-----+-----+-----+-----+-----+-----+-----+
| 10101|Chachapoyas|Chachapoyas|  Amazonas|  29171|  153.78|-6.2294|-77.8714|
| 10102|  Asuncion|Chachapoyas|  Amazonas|    288|   25.71|-6.0317|-77.7122|
| 10103|   Balsas|Chachapoyas|  Amazonas|   1644|  357.09|-6.8375|-78.0214|
| 10104|   Cheto|Chachapoyas|  Amazonas|    591|   56.97|-6.2558|-77.7003|
| 10105|Chiliquin|Chachapoyas|  Amazonas|    687|  143.43|-6.0778|-77.7392|
```

11.1.2 Limpieza de Dataframes

- df_CIE: Solo cambió el formato de nombre en las columnas

Código:

```
df_CIE = df_CIE.withColumnRenamed("CODIGO", "cod_enfermedad") \
               .withColumnRenamed("DESCRIPCION", "des_enfermedad")
```

Resultado:

```
+-----+-----+
|cod_enfermedad|  des_enfermedad|
+-----+-----+
|      Y21.4|Ahogamiento y sum...|
|      Y21.5|Ahogamiento y sum...|
|      Y21.6|Ahogamiento y sum...|
|      Y21.7|Ahogamiento y sum...|
|      Y21.8|Ahogamiento y sum...|
+-----+-----+
```

- df_Diabetes: Esta tabla se normalizará y se obtendrán las tablas de: Procedimiento, Resultado_Diagnostico, Medico y Paciente

Código:

- Para poner en formato la tabla df_Diabetes

```
from pyspark.sql.functions import col, row_number, monotonically_increasing_id
from pyspark.sql.window import Window
from pyspark.sql.functions import to_date, col
```

```
# Creamos una ventana sin partición ni orden específico
windowSpec = Window.orderBy(monotonically_increasing_id())
```

```
df_Diabetes_main = df_Diabetes.select(
    col("COD_DIAG").alias("cod_enfermedad"),
    col("ID_PACIENTE").alias("id_paciente"),
    col("IPRESS").alias("cod_institucion"),
    col("ID_MEDICO").alias("id_medico"),
    col("SERVICIO_HOSPITALARIO").alias("servicio_hospitalario"),
    col("ACTIVIDAD_HOSPITALARIA").alias("actividad_hospitalaria"),
    col("FECHA_MUESTRA").alias("fecha_muestra")
)
```

```
# Añadimos cod_diagnostico como número incremental
df_Diabetes_main = df_Diabetes_main.withColumn(
    "cod_diagnostico", row_number().over(windowSpec)
)
```

```
# Corregimos la fecha
df_Diabetes_main = df_Diabetes_main.withColumn(
    "fecha_muestra",
    to_date(col("fecha_muestra").cast("string"), "yyyyMMdd")
)
```

- Para crear la tabla Procedimiento

```
from pyspark.sql import Window
from pyspark.sql.functions import col, row_number, monotonically_increasing_id
```

```
# Tomamos ambos conjuntos de procedimientos
df_proc1 = df_Diabetes.select(col("PROCEDIMIENTO_1").alias("des_procedimiento")).distinct()
df_proc2 = df_Diabetes.select(col("PROCEDIMIENTO_2").alias("des_procedimiento")).distinct()
```

```
# Unimos, eliminamos duplicados y asignamos un código incremental
windowSpec = Window.orderBy(monotonically_increasing_id())
```

```
df_Procedimiento = (
    df_proc1
    .unionByName(df_proc2)
    .distinct()
    .withColumn("cod_procedimiento", row_number().over(windowSpec))
    .select("cod_procedimiento", "des_procedimiento")
)
```

- Para crear la tabla Resultado_Diagnostico


```

from pyspark.sql.functions import col
from pyspark.sql.functions import to_date, col

# Unimos df_Diabetes con df_Diabetes_main para obtener el cod_diagnostico
df_joined = df_Diabetes_main.join(
    df_Diabetes,
    ["id_paciente", "id_medico", "fecha_muestra"],
    "inner"
)

# Creamos df_res1 y df_res2 con los datos de resultados
df_res1 = df_joined.select(
    col("cod_diagnostico"),
    col("PROCEDIMIENTO_1").alias("des_procedimiento"),
    col("RESULTADO_1").alias("resultado"),
    col("UNIDADES_1").alias("unidades"),
    col("FEC_RESULTADO_1").alias("fecha_resultado")
)

df_res2 = df_joined.select(
    col("cod_diagnostico"),
    col("PROCEDIMIENTO_2").alias("des_procedimiento"),
    col("RESULTADO_2").alias("resultado"),
    col("UNIDADES_2").alias("unidades"),
    col("FEC_RESULTADO_2").alias("fecha_resultado")
)

# Unimos ambos conjuntos
df_Resultado_Procedimiento_temp = df_res1.unionByName(df_res2).distinct()

# Asociamos el código de procedimiento según el nombre
df_Resultado_Procedimiento = df_Resultado_Procedimiento_temp.join(
    df_Procedimiento,
    on=["des_procedimiento"],
    how="left"
).select(
    "cod_diagnostico",
    "cod_procedimiento",
    "resultado",
    "unidades",
    "fecha_resultado"
)

# Corregimos la fecha
df_Resultado_Procedimiento = df_Resultado_Procedimiento.withColumn(
    "fecha_resultado",
    to_date(col("fecha_resultado").cast("string"), "yyyyMMdd")
)

```

- Para crear la tabla Medico

```

df_Medico = df_Diabetes.select(
    col("ID_MEDICO").alias("id_medico"),
    col("EDAD_MEDICO").alias("edad_medico")
).distinct()

```

- Para crear la tabla Paciente

```
df_Paciente = df_Diabetes.select(
    col("ID_PACIENTE").alias("id_paciente"),
    col("EDAD_PACIENTE").alias("edad_paciente"),
    col("SEXO_PACIENTE").alias("sexo_paciente"),
).distinct()
```

Resultado:

- df_Diabetes

cod_enfermedad	id_paciente	cod_institucion	id_medico	servicio_hospitalario	actividad_hospitalaria	fecha_muestra	cod_diagnostico
E11.9 eJwzNDAwTDC0NDMxN...		CAP I MANANTAY eJwzNjA2MzE2NLY0N...		MEDICINA GENERAL	ATENCION MEDICA ...	2020-01-02	1
E13.9 eJwzNDAwMjAxdNTUxs...		P.M. ALAMEDA eJwzsjs1NDI2MjE2N...		MEDICINA GENERAL	ATENCION MEDICA ...	2020-01-02	2
E11.9 eJwzNDCwMlOwMDKzt...	H.III DANIEL ALCI...	eJwztlQwNjM1NTY3N...		ENDOCRINOLOGIA	ATENCION MEDICA ...	2020-01-02	3
E11.9 eJwzNDCwNDU3Mja1M...	CAP II OSCAR FERN...	eJwztlS0MDE1s3Q3N...		MEDICINA FAMILIAR...	ATENCION MEDICA ...	2020-01-02	4
E11.9 eJwzNDA0szQCQgMDM...		P.M. SAPOSOA eJwzNDS1MDO0MDEyM...		MEDICINA GENERAL	ATENCION MEDICA ...	2020-01-02	5

- df_Procedimiento

cod_procedimiento	des_procedimiento
1	DOSAJE DE GLUCOSA...
2	DOSAJE DE COLESTE...

- df_Resultado_Diagnostico

cod_diagnostico	cod_procedimiento	resultado	unidades	fecha_resultado
335027	1	99.0	mg/dL	2023-05-18
155766	1	114.98	mg/dL	2022-04-02
60665	1	144.0	mg/dL	2021-04-26
289348	1	114.2	mg/dL	2023-02-14
29517	1	91.0	mg/dL	2020-11-04
64224	1	149.0	mg/dL	2021-05-17
111879	1	250.0	mg/dL	2021-11-11
49653	1	186.0	mg/dL	2021-02-15
127578	1	97.45	mg/dL	2021-12-29
188596	1	89.0	mg/dL	2022-06-21

- df_Medico

id_medico	edad_medico
eJwzNDQ1tzQxNLYwN...	55
eJwzMjQwNTY0tbAwN...	31
eJwzMjAzMLMwNTU0s...	33
eJwzND03sDAyMTAwM...	52
eJwzNDMwNbQ0NDK2N...	46
eJwztDQzt7Q0MTY1N...	41
eJwzNLYwNjM1M7Mwt...	62
eJwzMjSwNDQwNzS0N...	32
eJwzNDY1MzYzNjc0M...	66
eJwzNDUztzQwMLIwN...	53

- df_Paciente

id_paciente	edad_paciente	sexo_paciente
eJwzNDY1NTQ0NDczN...	64	FEMENINO
eJwzNLQ0NbcwtbAwN...	51	FEMENINO
eJwzNDE2MzG0NLawM...	66	FEMENINO
eJwztDQ3NrS0BCIjC...	44	MASCULINO
eJwzNDYxMTGyMDI2N...	76	FEMENINO
eJwzNDY2tLAWNTE0M...	55	FEMENINO
eJwzNLY0MLAwNzA3N...	54	FEMENINO
eJwzNDIwMTM1sDQws...	48	MASCULINO
eJwzNLMwMLE0NDEwM...	47	MASCULINO
eJwzMjAyNLYwMjE0N...	38	MASCULINO

- df_Enf_Renal: Igual que para diabetes

Código:

```
from pyspark.sql.functions import col, row_number, monotonically_increasing_id
from pyspark.sql.window import Window
from pyspark.sql.functions import to_date, col

# Creamos una ventana sin partición ni orden específico
windowSpec = Window.orderBy(monotonically_increasing_id())

last_cod_diag = df_Diabetes_main.agg({"cod_diagnostico": "max"}).collect()[0][0]

df_Renal_main = df_Enf_Renal.select(
    col("COD_DIAG").alias("cod_enfermedad"),
    col("ID_PACIENTE").alias("id_paciente"),
    col("IPRESS").alias("cod_institucion"),
    col("ID_MEDICO").alias("id_medico"),
    col("SERVICIO_HOSPITALARIO").alias("servicio_hospitalario"),
    col("ACTIVIDAD_HOSPITALARIA").alias("actividad_hospitalaria"),
    col("FECHA_MUESTRA").alias("fecha_muestra")
)
```

```
)

# Añadimos cod_diagnostico como número incremental
df_Renal_main = df_Renal_main.withColumn(
    "cod_diagnostico", row_number().over(windowSpec) + last_cod_diag
)

# Corregimos la fecha
df_Renal_main = df_Renal_main.withColumn(
    "fecha_muestra",
    to_date(col("fecha_muestra").cast("string"), "yyyyMMdd")
)
```

Resultado:

cod_enfermedad	id_paciente	cod_institucion	id_medico	servicio_hospitalario	actividad_hospitalaria	fecha_muestra	cod_diagnostico
N18.3	eJwzNDU3NzI2NDGwN...	H.N. ALBERTO SABO...	eJwzNLQwNTM3Mzc3N...	NEFROLOGIA	ATENCION MEDICA ...	2020-01-02	509717
N18.9	eJwzNDUwMjQ2NjIzN...	H.II HUANUCO	eJwzNLUwNDQxMTYzN...	NEFROLOGIA	ATENCION MEDICA ...	2020-01-02	509718
N18.9	eJwzNDUwMDYwMjYyYt...	H.II HUANUCO	eJwzNLUwNDQxMTYzN...	NEFROLOGIA	ATENCION MEDICA ...	2020-01-02	509719
N18.9	eJwzNLG0MDYzMTZlZz...	H.II HUANUCO	eJwzNLUwNDQxMTYzN...	NEFROLOGIA	ATENCION MEDICA ...	2020-01-02	509720
N18.6	eJwzNDUwMDY2MjAxN...	H.II HUANUCO	eJwzNLUwNDQxMTYzN...	NEFROLOGIA	ATENCION MEDICA ...	2020-01-02	509721

- df_Hiperlipidemia: Igual que para diabetes

Código:

```
from pyspark.sql.functions import col, row_number, monotonically_increasing_id
from pyspark.sql.window import Window
from pyspark.sql.functions import to_date, col

# Creamos una ventana sin partición ni orden específico
windowSpec = Window.orderBy(monotonically_increasing_id())

last_cod_diag = df_Renal_main.agg({"cod_diagnostico": "max"}).collect()[0][0]

df_Hiperlipidemia_main = df_Hiperlipidemia.select(
    col("COD_DIAG").alias("cod_enfermedad"),
    col("ID_PACIENTE").alias("id_paciente"),
    col("IPRESS").alias("cod_institucion"),
    col("ID_MEDICO").alias("id_medico"),
    col("SERVICIO_HOSPITALARIO").alias("servicio_hospitalario"),
    col("ACTIVIDAD_HOSPITALARIA").alias("actividad_hospitalaria"),
    col("FECHA_MUESTRA").alias("fecha_muestra")
)

# Añadimos cod_diagnostico como número incremental
df_Hiperlipidemia_main = df_Hiperlipidemia_main.withColumn(
    "cod_diagnostico", row_number().over(windowSpec) + last_cod_diag
)

# Corregimos la fecha
df_Hiperlipidemia_main = df_Hiperlipidemia_main.withColumn(
    "fecha_muestra",
    to_date(col("fecha_muestra").cast("string"), "yyyyMMdd")
)
```

Resultado:

cod_enfermedad	id_paciente	cod_institucion	id_medico	servicio_hospitalario	actividad_hospitalaria	fecha_muestra	cod_diagnostico
	E78.2 eJwzNDQ2MzU2tLQwM...	POL. SAN LUIS eJwzNDCwNDG1NDYxM...		CARDIOLOGIA	ATENCION MEDICA ...	2020-01-02	625066
	E78.1 eJwzNLY0tDC1sDQ1N...	H. I LA ESPERANZA eJwzNLYwNzcXNzQxN...		MEDICINA INTERNA	ATENCION MEDICA ...	2020-01-02	625067
	E78.2 eJwzNjAwNlc0MjI1N...	H. II GUSTAVO LANA... eJwzNDU3NT6xMDczM...		PEDIATRIA	ATENCION MEDICA ...	2020-01-03	625068
	E78.5 eJwzsJQ3NrUwNTMxM...	H. II RENE TOCHE G... eJwzMjAwNTYwNDU3M...		MEDICINA GENERAL	ATENCION MEDICA ...	2020-01-03	625069
	E78.2 eJwzMjQxMTQzN7IwN...	CAP III METROPOLI... eJwzMjAwNTQwNLEwM...		MEDICINA GENERAL	ATENCION MEDICA ...	2020-01-03	625070

- df_Geodir: Se quitaron columnas no importantes y se cambió el nombre de las columnas

Codigo:

```
df_Geodir = df_Geodir.select(
    col("Ubigeo").alias("ubigeo"),
    col("Distrito").alias("distrito"),
    col("Provincia").alias("provincia"),
    col("Departamento").alias("departamento"),
    col("Poblacion").alias("poblacion")
)
```

Resultado:

ubigeo	distrito	provincia	departamento	poblacion
	10101	Chachapoyas	Chachapoyas	Amazonas
	10102	Asuncion	Chachapoyas	Amazonas
	10103	Balsas	Chachapoyas	Amazonas
	10104	Cheto	Chachapoyas	Amazonas
	10105	Chiliquin	Chachapoyas	Amazonas
	10106	Chuquibamba	Chachapoyas	Amazonas
	10107	Granada	Chachapoyas	Amazonas
	10108	Huancas	Chachapoyas	Amazonas
	10109	La Jalca	Chachapoyas	Amazonas
	10110	Leimebamba	Chachapoyas	Amazonas

- df_Ipress: Se quitaron columnas no importantes y se cambió el nombre de las columnas

Codigo:

```
from pyspark.sql import functions as F

df_Ipress = (
    df_Ipress
    .select(
        F.col("Código Único").alias("cod_institucion"),
        F.col("UBIGEO").alias("ubigeo"),
        F.col("Tipo").alias("tipo"),
        F.col("Nombre del establecimiento").alias("nombre"),
        F.col("Clasificación").alias("clasificacion"),
    )
)
```

```

        F.col("Institución").alias("institucion"),
        F.col("Red").alias("red")
    )
    .dropna(subset=["cod_institucion"])
    .dropDuplicates(["cod_institucion"])
)

```

Resultado:

cod_institucion	ubigeo	tipo	nombre	clasificacion	institucion	red
	10096 140308	ESTABLECIMIENTO D...	EL PUEBLITO PUESTOS DE SALUD ...	GOBIERNO REGIONAL	LAMBAYEQUE	
	10351 150116	ESTABLECIMIENTO D...	RENAN JAE OTTA G... CONSULTORIOS MEDI...	PRIVADO	NO PERTENECE A NI...	
	1090 190301	ESTABLECIMIENTO D...	MEZAPATA PUESTOS DE SALUD ...	GOBIERNO REGIONAL	OXAPAMPA	
	11078 140101	ESTABLECIMIENTO D...	SEGUNDO RAFAEL SA... CONSULTORIOS MEDI...	PRIVADO	NO PERTENECE A NI...	
	11332 150132	ESTABLECIMIENTO D...	PACPAC ROMAN WILB... CONSULTORIOS MEDI...	PRIVADO	NO PERTENECE A NI...	
	1159 190306	ESTABLECIMIENTO D...	PUERTO AGUACHINI PUESTOS DE SALUD ...	GOBIERNO REGIONAL	OXAPAMPA	
	11888 150131	ESTABLECIMIENTO D...	RICARDO ODRIA & A... CONSULTORIOS MEDI...	PRIVADO		
	12394 230101	ESTABLECIMIENTO D...	MEDICINA INTEGRAL... CONSULTORIOS MEDI...	PRIVADO	NO PERTENECE A NI...	
	12529 20508	ESTABLECIMIENTO D...	POLICLINICO HUANZALA	POLICLINICOS	PRIVADO	NO PERTENECE A NI...
	12847 150143	ESTABLECIMIENTO D...	PUESTO DE SALUD C... PUESTOS DE SALUD ...	MINSA	NO PERTENECE A NI...	

- Por último, unificamos todos los diagnósticos de diabetes, enfermedades renales e hiper en un solo df:

```

df_diagnostico = (
    df_Diabetes_main
    .unionByName(df_Renal_main)
    .unionByName(df_Hiperlipidemia_main)
)

```

cod_enfermedad	id_paciente	cod_institucion	id_medico	servicio_hospitalario	actividad_hospitalaria	fecha_muestra	cod_diagnostico
	E11.9 eJwzNDAwtdC0NDMxN...	CAP I MANANTAY eJwzNjA2MzE2NLY0N...		MEDICINA GENERAL	ATENCION MEDICA ...	2020-01-02	1
	E13.9 eJwzNDAwMjAxNTUxs...	P.M. ALAMEDA eJwzsj51NDI2MjE2N...		MEDICINA GENERAL	ATENCION MEDICA ...	2020-01-02	2
	E11.9 eJwzNDCwMLOWMDKzt...	H.III DANIEL ALCI... eJwztLQwNjM1NTY3N...		ENDOCRINOLOGIA	ATENCION MEDICA ...	2020-01-02	3
	E11.9 eJwzNDCwNDU3Mja1M...	CAP II OSCAR FERN... eJwztLS0MDE1s3jQ3N...		MEDICINA FAMILIAR...	ATENCION MEDICA ...	2020-01-02	4
	E11.9 eJwzNDA0szQCQgMDM...	P.M. SAPOSOA eJwzNDS1MD00MDEyM...		MEDICINA GENERAL	ATENCION MEDICA ...	2020-01-02	5
	E11.9 eJwzNLY0tDC1sDQ1N...	H.I LA ESPERANZA eJwzNLYwNzcXNzQxN...		MEDICINA INTERNA	ATENCION MEDICA ...	2020-01-02	6
	E11.9 eJwzNLY0WjM3NzE2M...	H.I LA ESPERANZA eJwzNLY0NzE3MjMz...		MEDICINA GENERAL	ATENCION MEDICA ...	2020-01-02	7
	E11.9 eJwzNLY0tjQ1NzYwN...	H.III SUAREZ-ANGAMOS eJwzNDA0ACJjAzNLU...		ENDOCRINOLOGIA	ATENCION MEDICA ...	2020-01-02	8
	E11.4 eJwzNDU0NzA0MDc0t...	H.II HUANUCO eJwzMjAwM7SwMLU0N...		MEDICINA INTERNA	ATENCION MEDICA ...	2020-01-02	9
	E11.9 eJwzMfUwMjYxsfA3s...	P.M. SAPOSOA eJwzNDS1MD00MDEyM...		MEDICINA GENERAL	ATENCION MEDICA ...	2020-01-02	10

11.2. Esquemas, particiones y propiedades de tablas

11.2.1 Exportando las tablas limpias como archivos csv:

Código:

```

# Ruta base
output_path = "/data/final/"

# Exportamos cada DataFrame
df_diagnostico.coalesce(1).write.mode("overwrite").option("header", "true").csv(output_path + "diagnostico")
df_paciente.coalesce(1).write.mode("overwrite").option("header", "true").csv(output_path + "paciente")
df_medico.coalesce(1).write.mode("overwrite").option("header", "true").csv(output_path + "medico")
df_resultado_procedimiento.coalesce(1).write.mode("overwrite").option("header", "true").csv(output_path + "resultado_procedimiento")

```

```
df_procedimiento.coalesce(1).write.mode("overwrite").option("header", "true").csv(output_path + "procedimiento")
df_ubigeo.coalesce(1).write.mode("overwrite").option("header", "true").csv(output_path + "ubigeo")
df_ipress.coalesce(1).write.mode("overwrite").option("header", "true").csv(output_path + "ipress")
df_CIE.coalesce(1).write.mode("overwrite").option("header", "true").csv(output_path + "cie")
```

Resultado:

Name >	Size >	Last Modified >	Owner >
cie	--	2025-10-12 20:42	zeppelin
diagnostico	--	2025-10-12 20:39	zeppelin
diagnostico_parquet	--	2025-10-12 20:50	zeppelin
ipress	--	2025-10-12 20:42	zeppelin
medico	--	2025-10-12 20:40	zeppelin
paciente	--	2025-10-12 20:39	zeppelin
procedimiento	--	2025-10-12 20:42	zeppelin
resultado_procedimiento	--	2025-10-12 20:41	zeppelin
ubigeo	--	2025-10-12 20:42	zeppelin

11.2.2. Exportando las tablas limpias como archivos Spark con parket:

- Particiones de data por año:

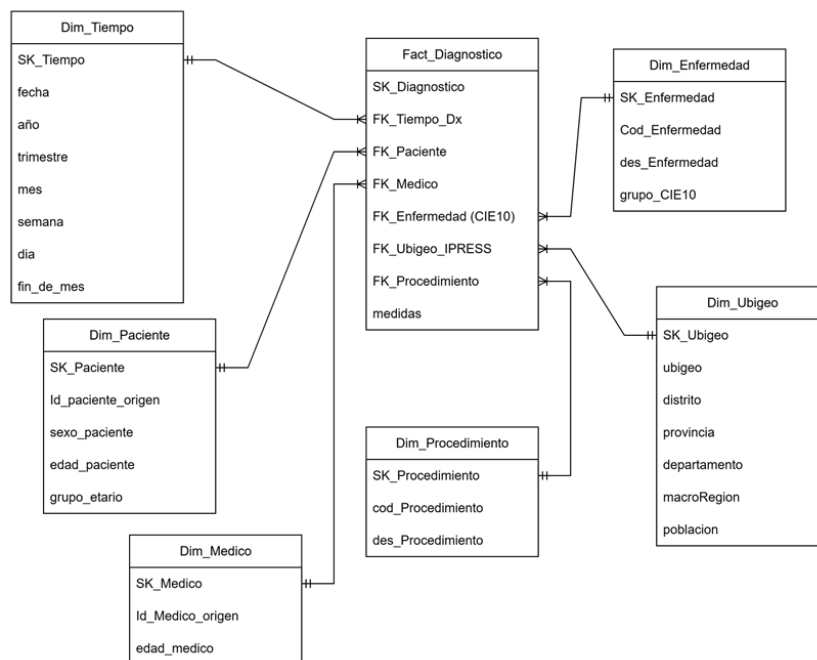
Name >	Size >	Last Modified >	Owner >	Group >	Permission
_SUCCESS	0.1 kB	2025-10-12 21:12	zeppelin	hdfs	-rw-r--r--
anio=2020	--	2025-10-12 21:12	zeppelin	hdfs	drwxr-xr-x
anio=2021	--	2025-10-12 20:51	zeppelin	hdfs	drwxr-xr-x
anio=2022	--	2025-10-12 21:12	zeppelin	hdfs	drwxr-xr-x
anio=2023	--	2025-10-12 20:51	zeppelin	hdfs	drwxr-xr-x
anio=2024	--	2025-10-12 20:51	zeppelin	hdfs	drwxr-xr-x

- Particiones de data por meses:

diagnostico_parquet > año=2020						Total: 12 files or folders	+ Select All	New Folder	Upload
Search in current directory...									
Name >	Size >	Last Modified >	Owner >	Group >	Permission				
↩									
mes=1	--	2025-10-12 21:12	zeppelin	hdfs	drwxr-xr-x				
mes=10	--	2025-10-12 21:12	zeppelin	hdfs	drwxr-xr-x				
mes=11	--	2025-10-12 21:12	zeppelin	hdfs	drwxr-xr-x				
mes=12	--	2025-10-12 21:12	zeppelin	hdfs	drwxr-xr-x				
mes=2	--	2025-10-12 21:12	zeppelin	hdfs	drwxr-xr-x				
mes=3	--	2025-10-12 21:12	zeppelin	hdfs	drwxr-xr-x				
mes=4	--	2025-10-12 21:12	zeppelin	hdfs	drwxr-xr-x				
mes=5	--	2025-10-12 21:12	zeppelin	hdfs	drwxr-xr-x				
mes=6	--	2025-10-12 21:12	zeppelin	hdfs	drwxr-xr-x				
mes=7	--	2025-10-12 21:12	zeppelin	hdfs	drwxr-xr-x				

TABLAS, CUBOS Y MODELOS OLAP

12.1. Diseño de tablas fact/dimension



12.1.1 Preparación inicial de data

Código:

- Creación de la tabla

```
df_fact = (
    df_diagnostico
    .join(df_paciente, "id_paciente", "left")
)
```



```

.join(df_resultado_procedimiento, "cod_diagnostico", "left")
.join(df_procedimiento, "cod_procedimiento", "left")
.join(df_ipress, "cod_institucion", "left")
.join(df_ubigeo, "ubigeo", "left")
.join(df_CIE, "cod_enfermedad", "left")
)

```

- Selección de columnas relevantes:

```

df_fact = df_fact.select(
    "cod_diagnostico",
    "id_paciente",
    "sexo_paciente",
    "grupo_etario",
    "cod_institucion",
    "nombre",
    "departamento",
    "red",
    "cod_enfermedad",
    "des_enfermedad",
    "des_procedimiento",
    "resultado",
    "unidades",
    "fecha_resultado"
)

```

- Limpieza final

```

%pyspark

df_fact = df_fact.dropna(subset=["resultado"])

```

12.1.2 Comparación Raw vs Curated

Código:

```

# RAW
df_diagnostico.write.mode("overwrite").parquet("/user/hive/warehouse/raw/diagnostico")

# CURATED
df_fact.write.mode("overwrite").partitionBy("departamento").parquet("/user/hive/warehouse/curated/fact_d
iagnostico")

```

12.2.1 Integración con Hive

A) Creación y visualización de bases de datos en Apache Spark con PySpark

En esta etapa, se muestra el proceso de creación y verificación de bases de datos dentro del metastore de Hive utilizando PySpark, con el propósito de estructurar las diferentes capas de almacenamiento del modelo de datos del proyecto. Estas capas se emplean para separar los datos en su forma cruda (raw) y en su forma depurada o transformada (curated), lo cual facilita la organización, trazabilidad y posterior análisis de la información en entornos Big Data.

```

"""

```

```

%pyspark

```

```
spark.sql("SHOW DATABASES").show()
"""
```

El comando %pyspark indica que el intérprete usado es PySpark dentro del entorno Zeppelin.

La instrucción spark.sql("SHOW DATABASES") ejecuta una sentencia SQL en el motor de Spark para listar las bases de datos registradas en el metastore.

```
+-----+
|databaseName|
+-----+
|      default|
|    foodmart|
+-----+
```


Inicialmente se observan dos bases de datos:

- default: creada por defecto por Spark.
- foodmart: base de datos de ejemplo utilizada para pruebas en entornos analíticos.

```
"""
```

```
%pyspark
```

```
from pyspark.sql import SparkSession
```

```
spark = (
    SparkSession.builder
        .appName("Proyecto_OLAP")
        .config("spark.sql.warehouse.dir", "/tmp/hive_warehouse") #  ruta donde sí puedes escribir
        .enableHiveSupport()
        .getOrCreate()
)
"""
```

En este bloque se inicializa una sesión de Spark denominada Proyecto_OLAP, la cual se conecta al metastore de Hive mediante el parámetro .enableHiveSupport().

El atributo .config("spark.sql.warehouse.dir", "/tmp/hive_warehouse") define la ruta del warehouse donde se almacenarán las tablas y bases creadas.

De esta manera, Spark queda configurado para actuar como un motor SQL distribuido con capacidad de persistir estructuras de datos tipo Hive.

```
"""
```

```
%pyspark
spark.sql("CREATE DATABASE IF NOT EXISTS raw_db")
spark.sql("CREATE DATABASE IF NOT EXISTS curated_db")
"""
```

Se crean dos bases de datos que representan las capas principales del flujo de datos:

- raw_db: almacena los datos en su estado original o sin procesar.
- curated_db: almacena los datos ya transformados, limpios y listos para consumo analítico.

El uso de IF NOT EXISTS garantiza que el proceso no genere errores si las bases ya existen en el metastore.

```
"""
```

```
%pyspark
spark.sql("SHOW DATABASES").show()
```

```
+-----+
|databaseName|
+-----+
|  curated_db|
+-----+
```

```
|      default |
|      foodmart |
|      raw_db |
+-----+
<<"">>
```

El resultado confirma la creación satisfactoria de las nuevas bases de datos raw_db y curated_db, las cuales se integran al catálogo del metastore de Hive y estarán disponibles para posteriores procesos ETL, modelado y análisis.

B) Carga de tablas en las bases de datos Hive con PySpark

Una vez creadas las bases de datos raw_db y curated_db en el metastore de Hive, se procede a la carga de los conjuntos de datos (DataFrames) hacia las tablas correspondientes.

Este paso tiene como finalidad persistir la información en formato estructurado dentro del entorno de Spark, permitiendo su reutilización en posteriores procesos de análisis, modelado o visualización.

```
<<"">>
%pyspark
df_diagnostico.write.mode("overwrite").saveAsTable("raw_db.diagnostico")
df_paciente.write.mode("overwrite").saveAsTable("raw_db.paciente")
df_medico.write.mode("overwrite").saveAsTable("raw_db.medico")
df_resultado_procedimiento.write.mode("overwrite").saveAsTable("raw_db.resultado_procedimiento")
df_procedimiento.write.mode("overwrite").saveAsTable("raw_db.procedimiento")
df_ipress.write.mode("overwrite").saveAsTable("raw_db.ipress")
df_CIE.write.mode("overwrite").saveAsTable("raw_db.cie")
df_ubigeo.write.mode("overwrite").saveAsTable("raw_db.ubigeo")
<<"">>
```

Cada línea del bloque escribe un DataFrame de PySpark en el catálogo de Hive, convirtiéndolo en una tabla permanente dentro de la base raw_db.

El método .write.mode("overwrite").saveAsTable() cumple tres funciones principales:

- write: indica que se va a escribir el contenido del DataFrame.
- mode("overwrite"): reemplaza el contenido existente de la tabla si ya fue creada anteriormente, asegurando que siempre se tenga la versión más reciente de los datos.
- saveAsTable("base.tabla"): guarda la tabla en el metastore de Hive con el nombre especificado.

Con este proceso, las tablas diagnostico, paciente, medico, resultado_procedimiento, procedimiento, ipress, cie y ubigeo quedan almacenadas de manera persistente en la base raw_db.

Estas tablas representan la capa de datos crudos (Raw Layer) del sistema analítico, en la cual se conserva la información tal como fue obtenida de las fuentes originales, sin transformaciones complejas.

```
<<"">>
%pyspark
df_fact.write.mode("overwrite").saveAsTable("curated_db.fact_diagnostico")
<<"">>
```

En este paso, el DataFrame df_fact, que contiene la información consolidada de diagnósticos y procedimientos, se almacena en la base curated_db bajo el nombre fact_diagnostico.

Esta tabla representa una tabla de hechos dentro del modelo analítico, propia de la capa curated, que concentra la información procesada y depurada.

El modo overwrite asegura que los datos se actualicen completamente cada vez que se ejecute el proceso.

La tabla curated_db.fact_diagnostico constituye el punto de partida para la construcción de reportes o análisis OLAP, ya que su estructura está diseñada para relacionarse con las dimensiones de pacientes, médicos, procedimientos y establecimientos.

```
<<"">>
from pyspark.sql import functions as F

df_fact = df_fact.withColumn("anio", F.year("fecha_resultado"))
df_fact = df_fact.withColumn("mes", F.month("fecha_resultado"))
<<"">>
```

Aquí se añaden dos columnas nuevas al DataFrame df_fact mediante las funciones de PySpark:

F.year("fecha_resultado"): extrae el año de la columna fecha_resultado.

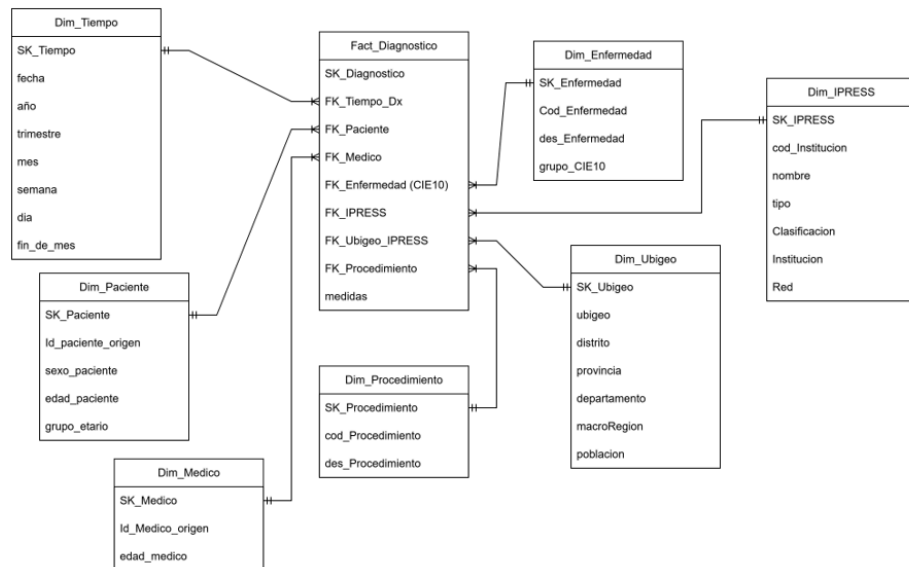
F.month("fecha_resultado"): extrae el mes de la misma columna.

De esta manera, se enriquecen los datos con variables temporales que facilitan el análisis por periodos (año y mes), lo cual es esencial para la creación de indicadores y visualizaciones temporales en tableros de control o modelos OLAP.

El DataFrame resultante posee ahora un esquema más analítico, que permite segmentar los resultados por dimensiones de tiempo.

Este paso constituye una parte clave del proceso de curación de datos, ya que transforma la información bruta en un formato más adecuado para consultas agregadas y análisis exploratorios.

12.2. Esquema estrella físico y diagramas



12.3. Creación del cubo OLAP inicial

- Hecho

La tabla `df_fact` representa el hecho principal del sistema analítico.

En este caso, el hecho es el diagnóstico médico realizado a un paciente, ya que es el evento que se puede medir y analizar.

- Dimensiones

Las dimensiones son los ejes por los que puedes analizar los hechos. En el código, se crea un cubo con 3 dimensiones principales:

Dimensión	Descripción	Ejemplo de uso
-----------	-------------	----------------

Grupo_Etario	Agrupar pacientes por rangos de edad (niño, joven, adulto, adulto mayor)	Analizar cómo varía el diagnóstico por grupo etario
Departamento	Ubicación geográfica del diagnóstico	Comparar resultados entre departamentos del país
Des_Enfermedad	Nombre o descripción de la enfermedad (por CIE)	Ver qué enfermedades son más frecuentes por región o edad
Mes y Año	Dimensión temporal para analizar tendencias en el tiempo	Comparar evolución de enfermedades por mes/año

DASHBOARD PRELIMINAR Y CONSUMO

13.1. Implementación: Power BI

13.1.1 Fórmulas utilizadas:

```

cant_diagnosticos = SUM(Cubo_nuevo[cantidad_diagnosticos])

1 Casos_cn_complicacion_% =
2 VAR CasosConComp =
3     CALCULATE(
4         SUM(Cubo_nuevo[cantidad_diagnosticos]),
5         FILTER(
6             Cubo_nuevo,
7             NOT CONTAINSSTRING(UPPER(Cubo_nuevo[des_enfermedad]), "SIN")
8         )
9     )
10 VAR TotalCasos =
11     SUM(Cubo_nuevo[cantidad_diagnosticos])
12
13 RETURN
14 DIVIDE(CasosConComp, TotalCasos, 0) * 100

```

```

ly_prom_resultado % =
VAR AnioActual = SELECTEDVALUE ( Cubo_nuevo[anio] )
VAR PromActual = AVERAGE ( Cubo_nuevo[promedio_resultado] )
VAR PromAnterior =
    CALCULATE (
        AVERAGE ( Cubo_nuevo[promedio_resultado] ),
        FILTER (
            ALL ( Cubo_nuevo ),
            Cubo_nuevo[anio] = AnioActual - 1
        )
    )
RETURN
DIVIDE ( PromActual - PromAnterior, PromAnterior )

prom_resultado = AVERAGE(Cubo_nuevo[promedio_resultado])

Resultado x Diagnóstico =
DIVIDE(
    SUMX(
        Cubo_nuevo,
        Cubo_nuevo[cantidad_diagnosticos] * Cubo_nuevo[promedio_resultado]
    ),
    [cant_diagnosticos]
)

tasa_diagnosticos =
DIVIDE(
    [cant_diagnosticos],
    CALCULATE( [cant_diagnosticos], ALLSELECTED() ),
    0
) * 100

```

13.2. Capturas, archivo Power BI

Informe epidemiológico EsSalud

Resultado Promedio

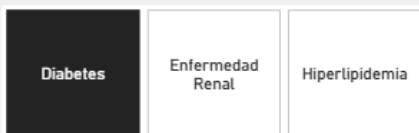
169.20

Total de diagnósticos

339 mil

Año

2023



Promedio de Resultado x Diagnóstico

168.17

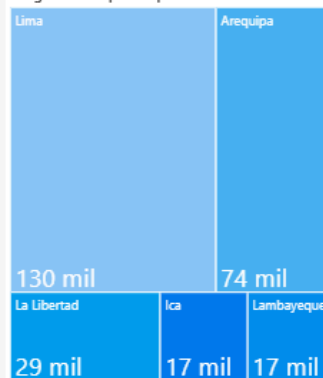
Porcentaje de Casos con Complicaciones

23.03

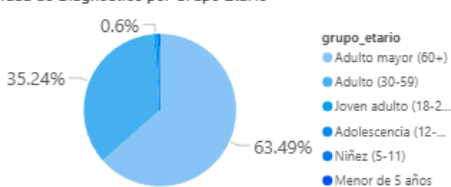
Diagnósticos x Mes



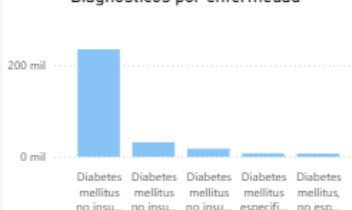
Diagnósticos por Departamento



Tasa de Diagnostico por Grupo Etario



Diagnósticos por enfermedad



Variación porcentual de resultados respecto al año anterior

33.6%