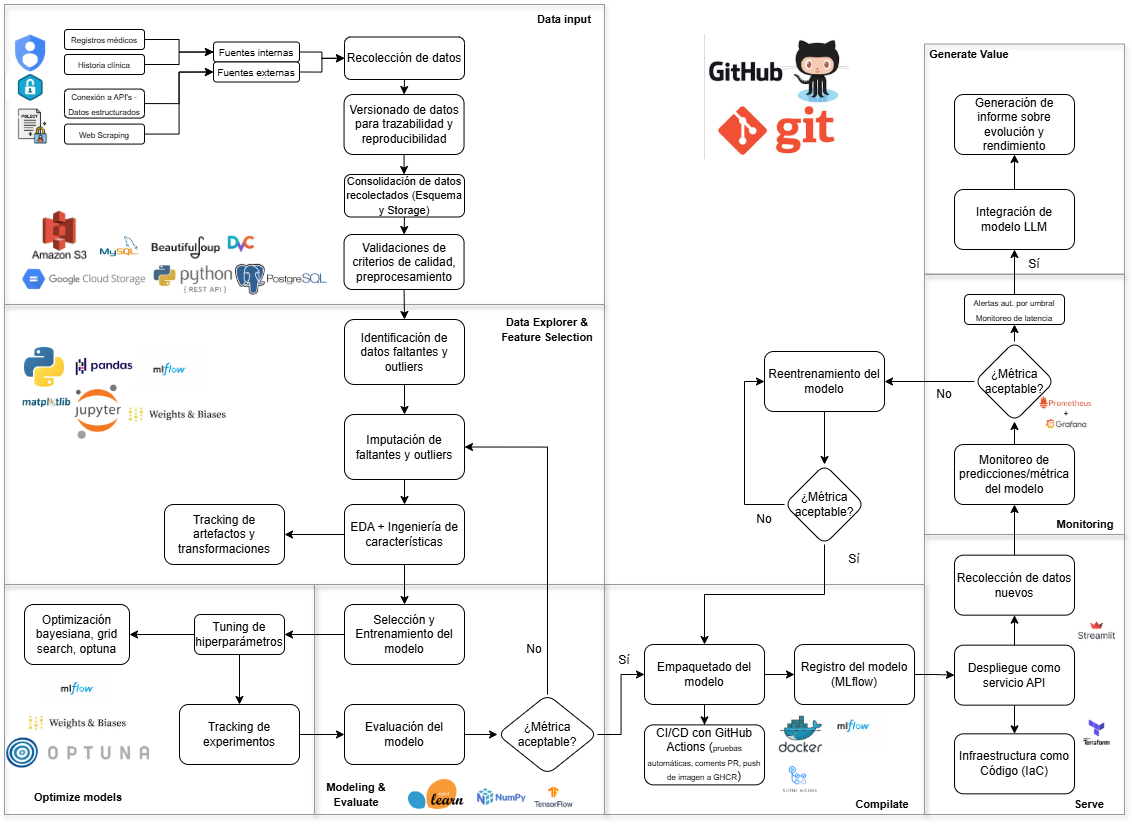
**Diseño**

1. **Diseño del pipeline de MLOps**



1. **Etapas del pipeline**
   1. **Etapa de Data input**
      1. **Recolección y versionado de datos:** En esta etapa se recopila información clínica relevante a partir de fuentes internas, como historias clínicas electrónicas y registros médicos que reposan en MySQL o PostgreSQL, y fuentes externas, como APIs estructuradas o técnicas de web scraping. A esta información se le realiza un proceso de versionado de datos, el cual permite garantizar la trazabilidad y reproducibilidad del flujo de trabajo. Esto asegura que los datos utilizados en etapas posteriores puedan ser auditados o replicados en futuras ejecuciones del modelo. El versionado se puede realizar mediante herramientas como DVC o sistemas de control integrados.
      2. **Consolidación y validación de los datos recolectados:** Una vez recolectados, los datos se integran bajo un esquema estructurado común y se almacenan en sistemas que puede ser Amazon S3, Google Cloud Storage. En esta etapa se realiza una validación de criterios de calidad de datos, la cual contempla aspectos como completitud, unicidad, consistencia, validez, exactitud y puntualidad. Estos criterios aseguran que los datos sean confiables y aptos para su uso posterior. Adicionalmente, se lleva a cabo un proceso de preprocesamiento inicial, corrigiendo formatos, transformaciones básicas y detección de errores, con el objetivo de entregar un conjunto de datos limpio y estructurado para el análisis exploratorio y modelado.
   2. **Etapa de Data Explorer & Feature Selection**
      1. **Identificación de datos faltantes y outliers:** Se detectan valores atípicos y datos ausentes utilizando herramientas como pandas, matplotlib y Jupyter Notebook, lo cual permite entender la calidad y distribución de las variables.
      2. **Imputación de faltantes y outliers:** Se aplican técnicas de imputación o eliminación, según el tipo de variable y su impacto, para asegurar que el conjunto de datos esté completo y sea robusto para el entrenamiento posterior.
      3. **EDA + Ingeniería de características:** Se realiza un análisis exploratorio de datos (EDA) para comprender correlaciones, tendencias y relaciones entre variables. En paralelo, se construyen nuevas variables a partir de las existentes para mejorar la capacidad predictiva del modelo.
      4. **Tracking de artefactos y transformaciones:** Con herramientas como MLflow y Weights & Biases, se lleva registro de las transformaciones realizadas, artefactos generados y configuraciones utilizadas. Esto garantiza trazabilidad, reproducibilidad y gestión eficiente del ciclo de vida de los datos.
   3. **Etapa de Modeling y Evaluate – Optimize Models**
      1. **Selección y entrenamiento del modelo:** Se elige el tipo de modelo (por ejemplo, regresión, árboles, redes neuronales) para selección multiclase y se entrena con los datos ya procesados. Se pueden utilizar herramientas como Scikit-learn, TensorFlow o NumPy según el tipo de modelo y complejidad.
      2. **Tuning de hiperparámetros:** Se ajustan los parámetros del modelo para maximizar su desempeño. Aquí se aplican técnicas como búsqueda en malla (Grid Search), búsqueda aleatoria o optimización bayesiana con herramientas como Optuna, Weights & Biases y MLflow.
      3. **Tracking de experimentos:** Se registra el desempeño de cada configuración del modelo y sus hiperparámetros, lo que permite compararlos fácilmente y mantener trazabilidad. Se usan plataformas como Weights & Biases o MLflow para esta gestión.
      4. **Evaluación del modelo:** Se mide el desempeño del modelo con métricas específicas (precisión, recall, F1, etc.), aplicadas al conjunto de validación para verificar su capacidad de generalización y comparando su capacidad predictiva para enfermedades
      5. **¿Métrica aceptable?:** Si el rendimiento del modelo cumple los criterios establecidos, se continúa al siguiente paso del pipeline. Si no, se retrocede a fases anteriores como la imputación de outliers y la ingeniería de características para revisar posibles ajustes en los datos.
   4. **Etapa de Compilate:** Una vez que el modelo ha alcanzado una métrica de rendimiento aceptable, se procede con esta etapa para asegurar su empaquetado, trazabilidad y automatización del despliegue:
      1. **Empaquetado del modelo:** El modelo validado se encapsula en un entorno reproducible, generalmente dentro de una imagen Docker. Este paso permite garantizar que el modelo se ejecute de la misma forma en diferentes entornos (local, testing, producción).
      2. **CI/CD con GitHub Actions:** Se automatizan las tareas de integración y despliegue continuo (CI/CD); Se ejecutan pruebas automáticas para validar que los cambios no rompan la funcionalidad; Se añaden comentarios automáticos en los PR para informar del estado del pipeline.
      3. **Registro del modelo (MLflow):** Finalmente, el modelo entrenado y aprobado se registra en MLflow, quedando documentado su historial de entrenamiento, métricas, artefactos y versiones. Esto facilita su seguimiento, comparación y reutilización futura.
   5. **Etapa de Serve**
      1. **Despliegue como servicio API:** El modelo es expuesto a través de una interfaz accesible por HTTP, generalmente como un endpoint REST. Esto permite que otros sistemas o usuarios realicen predicciones en tiempo real, herramientas como Streamlit permiten construir interfaces visuales interactivas para facilitar el uso por parte del equipo clínico u otros usuarios.
      2. **Infraestructura como Código (IaC):** Se utiliza IaC, como Terraform, para aprovisionar y gestionar automáticamente los recursos de infraestructura (servidores, redes, almacenamiento). Esto asegura un entorno reproducible, escalable y fácilmente modificable.
      3. **Recolección de nuevos datos:** En esta etapa se recogen nuevos registros de pacientes, ya sea provenientes del uso real del sistema, lo cual alimenta el proceso de actualización del modelo.
   6. **Etapa de Monitoring**
      1. **Monitoreo de predicciones / métricas del modelo:** Se realiza un seguimiento constante del rendimiento del modelo usando métricas como precisión, recall, F1-score, tasa de error o distribución de clases predichas; También se puede monitorear el input para detectar cambios significativos (data drift o concept drift).
      2. **¿Métrica aceptable?:** Se evalúan las métricas monitoreadas en tiempo real frente a umbrales predefinidos. Si la métrica es aceptable, el flujo continúa hacia la integración con modelos LLM y generación de valor. Si no es aceptable, se activa un proceso de reentrenamiento automático.
      3. **Alertas automáticas y monitoreo de latencia:** Con herramientas como Prometheus y Grafana, se implementan:
         1. Alertas automáticas por umbral para métricas clave.
         2. Monitoreo de latencia para asegurar respuestas rápidas del modelo en producción.
      4. **Loop de reentrenamiento:** Si las métricas no mejoran tras el reentrenamiento, se mantiene el ciclo hasta lograr una versión del modelo que supere los criterios establecidos. Si el modelo mejora, se pasa a la etapa de compilación (Compilate) para empaquetar y registrar la nueva versión.
   7. **Etapa de Generate Value:** Esta etapa representa el componente final del ciclo MLOps, orientado a transformar los resultados del modelo en valor real para los usuarios o stakeholders, agregando una capa de interpretación inteligente y documentación automatizada, a continuación se trata de detallar esta etapa con la finalidad de aclarar la observación hecha durante la retroalimentación del entregable 1 sobre este punto:
      1. **Condición de entrada: métricas aceptables:** Una vez que el modelo ha sido validado como óptimo desde la etapa de Monitoring, se permite su paso a esta fase. Esto garantiza que la solución es confiable y está lista para aportar valor.
      2. **Integración de Modelo LLM (Large Language Model):** Aquí se incorpora un modelo de lenguaje de gran escala, como GPT, para agregar capacidades de interpretación automática e interacción natural con el modelo. Algunas funciones clave del LLM en esta etapa incluyen:
         1. Interpretación contextual de resultados: El modelo LLM puede explicar los resultados de las predicciones de forma comprensible para usuarios no técnicos, como médicos o personal administrativo.
         2. Generación de explicaciones personalizadas: Dado un conjunto de predicciones o estadísticas, el modelo puede responder preguntas en lenguaje natural o redactar conclusiones automatizadas. Por ejemplo: “El 75% de los pacientes con frecuencia cardíaca mayor a 120 fueron clasificados como ENFERMEDAD AGUDA, lo que representa un incremento del 12% respecto a la semana anterior.”
         3. Traducción de métricas en hallazgos clínicos o recomendaciones: Por ejemplo, convertir una disminución en la precisión en una advertencia como “Podría haber un cambio en la calidad de los datos. Se recomienda revisar la fuente de registros clínicos.”
         4. Automatización de reportes ejecutivos o clínicos: Generación de documentos que combinan datos con lenguaje interpretativo, listos para entregarse a distintas áreas.

**Justificación técnica del uso del LLM**

Aunque el modelo base produce categorías limitadas (como NO ENFERMO, ENFERMEDAD LEVE, etc.), el modelo LLM agrega valor mediante su capacidad de interpretación, generación de lenguaje natural y síntesis de información. Esto permite:

* + - 1. Aumentar la accesibilidad del modelo a públicos no técnicos.
      2. Reducir la carga manual de interpretación de resultados por parte del equipo técnico.
      3. Facilitar la documentación continua de la evolución del modelo.
    1. **Generación automática de informes sobre evolución y rendimiento**

Con el apoyo del modelo LLM y los datos monitoreados, se generan automáticamente:

* + - 1. Reportes de evolución del modelo (precisión, F1-score, recall, etc.).
      2. Resúmenes del comportamiento del modelo por períodos de tiempo.
      3. Alertas explicadas en lenguaje natural si se detectan cambios relevantes o degradación en el desempeño.

1. **Restricciones, limitaciones y tipos de datos**

|  |  |  |  |
| --- | --- | --- | --- |
| **Fases** | **Restricciones** | **Limitaciones** | **Tipos de datos** |
| **Etapa Data Input** | | | |
| **Recolección de datos**. | Privacidad de los datos, restricciones legales, burocracia institucional | Poca disponibilidad de datos de enfermedades huérfanas, bajo nivel de digitalización de las entidades, | Datos estructurados, no estructurados |
| **Versionado de datos para trazabilidad y reproducibilidad** | Control de acceso a repositorios  Uso obligatorio de herramientas como DVC  Integridad de versiones | Requiere almacenamiento adicional Puede aumentar la complejidad operativa | Datos tabulares, históricos, CSV, JSON |
| **Consolidación de datos recolectados** | Estandarización de esquemas, validaciones de calidad | Problemas de interoperabilidad entre fuentes, errores de mapping entre campos de esquemas distintos | Datos estructurados provenientes de MySQL, PostgreSQL, APIs, S3, GCS |
| **Validaciones de criterios de calidad, preprocesamiento** | Calidad heterogénea provenientes de varias fuentes | Lograr consensos con entidades sobre el significado de un dato, presencia de errores, formatos inconsistentes, ambigüedades | Variables numéricas, categóricas, texto libre. |
| **Etapa de Data Explorer & Feature Selection** | | | |
| **Identificación e imputación de faltantes, outliers** | Restricciones de imputación por sensibilidad en el dato, sesgos | Imputación de faltantes o outliers sin contextos clínicos completos, |  |
| **EDA + Ingeniería de características** | Uso de datos anonimizados, restricciones de interpretación en datos sensibles, preprocesamiento obligatorio antes de aplicar modelos | Alto volumen puede limitar exploración visual, riesgo de overfitting con demasiadas variables | Variables numéricas, categóricas, temporales, derivadas |
| **Tracking de artefactos y transformaciones** | Registro obligatorio en herramientas como MLflow o W&B, reproducibilidad de transformaciones, nombres únicos para artefactos | Requiere integración con múltiples herramientas |  |
| **Etapa de Modeling y Evaluate – Optimize models** | | | |
| **Selección y entrenamiento del modelo** | Desequilibro de las clases (enfermedades normales y huérfanas) | Insuficiencia de datos etiquetados para diagnóstico de enfermedades huérfanas |  |
| **Tuning de hiperparámetros** | Definir espacio de búsqueda permitido, limitar número de combinaciones para evitar consumo excesivo de recursos | Alto costo computacional, tiempo de búsqueda elevado para modelos complejos | Valores numéricos y categóricos de parámetros del modelo |
| **Tracking de experimentos** | Registro completo y coherente de ejecuciones, asociar cada experimento con versión de código, datos y parámetros | Posible pérdida de trazabilidad sin uso riguroso de herramientas, dificultad para comparar múltiples ejecuciones |  |
| **Evaluación del modelo** | Métricas específicas por enfermedad normal y enfermedad huérfana, buen rendimiento del modelo para una de las dos clases | Métrica genérica como el accuracy no es la más conveniente, |  |
| **Etapa de Compilate** | | | |
| **Empaquetado del modelo** | Si el modelo se embebe en otro servicio la compatibilidad puede ser restrictiva, regulaciones sobre software medico si es diagnóstico asistido |  | .pkl |
| **CI/CD con Github Actions** | Ramas protegidas, validaciones automáticas antes de merge | Fallos en pipelines pueden bloquear despliegue, conocimientos en YAML y testing | Código fuente, archivos Dockerfile, YAML |
| **Registro del modelo** | Regulatorias al almacenar datos sensibles | Para escalar el modelo pueden haber dificultades si no se cuenta con infraestructura solida | Metadatos del entorno, versiones |
| **Etapa de Serve** | | | |
| **Despliegue como servicio API** | Validaciones clínicas previas, cumplimiento de normatividad en uso de datos y sector salud | Dependencia de infraestructura tech hospitalaria para integrar a sistemas reales |  |
| **Infraestructura como Codigo** | Control de cambios versionado, validaciones previas a despliegue, uso de variables seguras (secretos) | Dependencia de proveedores cloud, errores en scripts pueden causar fallos críticos en el entorno |  |
| **Recolección de datos nuevos** | Garantizar que los nuevos datos sean anonimizados | Frecuencia de datos nuevos |  |
| **Etapa de Monitoring** | | | |
| **Monitoreo de predicciones/métrica del modelo** | Recopilación continua de logs y métricas, umbrales definidos previamente para alertas, respeto por la privacidad del dato monitoreado | Latencia en detección de fallos, falta de métricas adecuadas puede limitar la evaluación en tiempo real | Los representados por las métricas de rendimiento |
| **Loop de reentrenamiento del modelo** | Validaciones médicas son obligatorias | Pocos datos nuevos para el reentrenamiento del modelo, costos computacionales |  |
| **Etapa de Generate Value** | | | |
| **Integración de modelo LLM (como apoyo, NO COMO DIAGNÓSTICO)** | LLMs deben evitar generar información médica incorrecta o no verificada (“alucinaciones”). Debe haber trazabilidad de lo generado por el modelo LLM | Alto consumo de recursos (memoria, procesamiento). | Informes, texto. |
| **Generación de informe sobre rendimiento y evolución del modelo** | Los reportes deben excluir información personal sensible. | Debe contar con Supervisión humana experta en el dominio y en modelamiento | Informes, texto. |

**Desarrollo.**

1. **¿De qué fuentes provienen los datos y cómo se manejan?**

**Los datos provienen de múltiples fuentes clínicas y externas, tales como**

* Historias clínicas electrónicas con información estructurada de pacientes como edad, sexo, antecedentes, diagnósticos previos, resultados de laboratorio, y las notas tomadas por los médicos durante las citas médicas
* Formularios clínicos o autodeclaraciones de síntomas donde los pacientes describen en texto libre cómo se sienten.
* Bases públicas o compartidas de enfermedades raras
* Datos abiertos del sector salud.
* APIs médicas y scraping para recolectar información complementaria sobre síntomas, prevalencias y criterios diagnósticos.

**Manejo de los datos**

* Preprocesamiento estructurado con la codificación de variables categóricas, normalización de valores numéricos.
* Procesamiento de texto (NLP), conversión de síntomas expresados en lenguaje natural a vectores numéricos y datos estructurados
* Imputación de datos faltantes y tratamiento de outliers clínicos.
* Validación de calidad y consistencia antes de alimentar el modelo.
* Anonimización y cumplimiento legal para proteger los datos sensibles del paciente.

**Seguridad y gobernanza de los datos**

* Se implementan mecanismos de seguridad y privacidad desde la etapa de adquisición, como el uso de protocolos de cifrado para la transmisión de datos y sistemas de autenticación para el acceso controlado a bases clínicas.
* El flujo incorpora controles de gobernanza sobre quién puede acceder, modificar o visualizar los datos, siguiendo políticas institucionales.
* Se integran herramientas de versionado como DVC que permiten rastrear cambios y mantener auditoría de los datos procesados.

1. **¿Qué tipos de modelos de ML se puede usar?**

Dado que el objetivo es clasificar si un paciente puede tener o no una enfermedad (común o huérfana), se recomienda usar modelos supervisados de clasificación, como: Random forest, XGBoost, Regresión Logística o Redes Neuronales Multicapa (MLP) aunque esta última muy limitada por la cantidad de datos que se puedan disponer

1. **¿Cómo se van a validar/testear?**

Para validar y testear el modelo predictivo, se implementará una estrategia de validación cruzada estratificada, que permite evaluar su desempeño asegurando una distribución representativa de enfermedades comunes y huérfanas en cada subconjunto. Esta técnica resulta esencial para garantizar que el modelo generalice adecuadamente en escenarios con desbalance de clases. Asimismo, se utilizarán métricas especializadas como el F1-score, la curva ROC-AUC, sensibilidad (recall), con el fin de capturar tanto la capacidad del modelo para detectar casos positivos como para evitar falsos diagnósticos.

**Producción**

1. **¿Cómo se va a desplegar la solución?**

La solución será desplegada mediante un servicio web basado en API, que permitirá integrarla fácilmente con plataformas clínicas o aplicaciones utilizadas por profesionales de la salud. El modelo será empaquetado en un contenedor utilizando herramientas como Docker, lo que asegura su portabilidad y reproducibilidad en distintos entornos, ya sea en servidores hospitalarios o en infraestructura en la nube (según sea el caso). Este despliegue garantizará que el sistema esté disponible para recibir síntomas como entrada y entregar predicciones en tiempo real, facilitando su uso clínico en escenarios de diagnóstico asistido (importante este último).

1. **¿Se necesita algún tipo de monitoreo para esta aplicación?**

Sí, es indispensable implementar un sistema de monitoreo continuo del modelo una vez esté en producción. Este monitoreo está detallado en el pipeline y permitirá detectar posibles caídas en el rendimiento predictivo, cambios en la distribución de los datos de entrada (data drift) o el surgimiento de nuevos patrones clínicos que el modelo no haya visto durante el entrenamiento. Además, se registrarán métricas clave como las descritas previamente además de los tiempos de respuesta, lo anterior servirá para generar alertas si estas métricas se desvían de los umbrales definidos por los equipos del dominio y de datos. El monitoreo también incluirá trazabilidad de las predicciones y el registro de logs para auditar el comportamiento del sistema ante cada solicitud.

1. **¿Pueden existir nuevos datos necesarios quizás para re-entrenar el modelo?**

Sí pueden existir nuevos datos que sirvan para reentrenar el modelo, en un entorno clínico real es esperable que se generen constantemente nuevos datos de pacientes, especialmente en el caso de enfermedades huérfanas que están en continuo estudio. Estos datos, una vez validados y anonimizados, pueden alimentar el pipeline de reentrenamiento del modelo para mejorar su precisión y adaptabilidad.

1. **Supuestos**
   1. **Acceso desde entornos locales con recursos limitados**
      1. Se asume que algunos médicos utilizarán el modelo desde computadores personales o institucionales con capacidades básicas (por ejemplo, CPU moderno, al menos 4 GB de RAM).
      2. Implicación: El modelo puede ser ejecutado localmente en un contenedor Docker liviano o, alternativamente, mediante solicitudes HTTP hacia una API expuesta en un servidor o nube, garantizando accesibilidad con baja latencia.
   2. **Despliegue modular y portable**
      1. El modelo se empaqueta en contenedores Docker, asegurando portabilidad entre entornos locales, servidores hospitalarios o servicios en la nube.
      2. Implicación: No se requiere una infraestructura especializada; basta con que el entorno local tenga Docker instalado.
   3. **Disponibilidad de infraestructura cloud (cuando se requiera)**
      1. Se asume que el hospital o institución dispone de acceso a servicios como AWS, GCP o Azure en caso de despliegues en la nube.
      2. Implicación: Se utilizará Terraform (IaC) para aprovisionar infraestructura automáticamente y mantener consistencia en entornos.
   4. **Conectividad y seguridad**
      1. Se parte del supuesto de que las estaciones locales tienen acceso a la red institucional (o internet, si se usa nube) y se implementarán medidas básicas de autenticación y cifrado.
      2. Implicación: El modelo puede recibir datos mediante solicitudes seguras (por ejemplo, HTTPS) y estar detrás de gateways o VPN institucionales
   5. **Disponibilidad de herramientas de trazabilidad**
      1. Se asume que los entornos de desarrollo y operación cuentan con herramientas como DVC, MLflow y Weights & Biases para control de versiones, artefactos y experimentos.
      2. Implicación: Toda transformación, entrenamiento o despliegue será registrado y trazable, incluso desde entornos descentralizados.
   6. **Acceso a los datos históricos y actualizados**
      1. Se asume que se podrá acceder a registros históricos anonimizados y que los nuevos datos pueden ser recolectados a través del uso del sistema por los médicos.
      2. Implicación: Los datos alimentarán periódicamente el proceso de reentrenamiento controlado por el equipo de ciencia de datos.
   7. **Supervisión médica garantizada en ciclo de reentrenamiento**
      1. Se parte del supuesto de que todo reentrenamiento será validado clínicamente antes de su reempaquetado y redeploy.
      2. Implicación: No se permite que el modelo se actualice sin aprobación médica o revisión de expertos.
   8. **Dependencia de la infraestructura TI hospitalaria**
      1. Se reconoce que la integración final del sistema puede estar condicionada por las capacidades tecnológicas del hospital (acceso a sistemas clínicos, políticas de TI, compatibilidades).
      2. Implicación: El pipeline prevé integración por APIs RESTful y monitoreo con herramientas estándar como Prometheus y Grafana.
   9. **Privacidad y gobernanza de datos**
      1. Se asume la existencia de normativas y acuerdos sobre el uso de datos anonimizados y gobernanza institucional que permiten la implementación de mecanismos de trazabilidad, cifrado y control de acceso.
      2. Implicación: El pipeline incluye en cada etapa mecanismos de protección de datos (versión, control de acceso, anonimización).
   10. **Disponibilidad de soporte técnico especializado bajo demanda**
       1. Se asume que el equipo de ciencia de datos, infraestructura o MLOps de la institución o del proveedor tiene la capacidad técnica para intervenir y dar soporte en caso de fallas, errores o procesos complejos como el reentrenamiento manual, auditorías o ajustes de configuración.
       2. Implicación: Esto permite delegar tareas críticas (como actualización de artefactos, resolución de errores de despliegue, validación de integridad del modelo o restauración de backups) a un equipo capacitado, garantizando continuidad operativa y confianza en el sistema ante eventos no controlados desde el entorno local del médico.