Universidad de los Andes > FACULTAD DE INGENIERÍA Y CIENCIAS APLICADAS

PROYECTO DE TÍTULO 1
Hito 1

Investigación sobre aspectos teóricos y modelado de procesos y datos

Alumno:
Gianfranco Bobadilla Riquelme

Profesor guía:
Claudio Álvarez.

Carrera:
Ingeniería Civil.

Octubre 5 de 2025

Índice

1. Introducción

El presente trabajo aborda la integración de factores humanos en ciberseguridad mediante la generación automatizada de reportes individuales de ciberhigiene. Partiendo de la necesidad de fortalecer las prácticas y la conciencia de los usuarios en instituciones que manejan información crítica (por ejemplo, hospitales y puertos), se propone un proceso que transforma datos cuantitativos en informes cualitativos claros, accionables y adaptados al perfil de cada encuestado. Estos reportes funcionan como una "receta médica" de ciberhigiene respaldadas por evidencia contenida en una base de conocimiento preprocesada.

El proyecto se ubica dentro de la plataforma *HFShield* y se focaliza en el componente *Relay*, en el módulo de generación de reportes individuales. De esta forma, se desarrolla una revisión de conceptos sobre factores humanos y ciberhigiene, distintas alternativas arquitectónicas para integrar modelos de lenguaje en el flujo de generación de informes, comparativa de diversos modelos de lenguaje y propuestas de un modelo de procesos y un esquema de datos para la *API* que soportará la entrega de reportes en la plataforma. El documento se centra en el diseño y las decisiones previas a la implementación del proyecto completo.

1.1 Objetivos generales

- Analizar los factores humanos en ciberseguridad y ciberhigiene, integrando enfoques bibliográficos y técnicos.
- Diseñar un modelo de proceso apoyado en modelos de lenguaje para la generación de reportes individuales personalizados a partir de datos cuantitativos.

1.2 Objetivos particulares

- Investigar enfoques arquitectónicos para el uso de modelos de lenguaje en el análisis cualitativo de datos cuantitativos y construir una tabla comparativa de modelos.
- Proponer un modelo de proceso y un esquema de datos para sustentar el servicio (API) de generación de reportes individuales personalizados.

2. Revisión bibliográfica

En esta sección se exploran los conceptos y definiciones clave para realizar los objetivos de este informe. Principalmente se exploran los factores humanos, la ciberseguridad y la ciberhigiene, usando referencias en bibliografía relacionada.

2.1 Ciberseguridad

La ciberseguridad es el conjunto de prácticas, tecnologías y políticas destinadas a proteger los sistemas informáticos, redes, aplicaciones y datos frente a accesos no autorizados, robos de información o daños. En la actualidad, la ciberseguridad se ha convertido en un pilar fundamental tanto para individuos como para organizaciones, ya que los ciberataques pueden ocasionar pérdidas económicas, afectar la reputación de una empresa e incluso poner en riesgo la seguridad nacional.

Se entiende por ciberseguridad no solo la instalación de programas de protección, sino también la implementación de medidas de prevención, detección y respuesta ante incidentes. Como señala IBM, *"cybersecurity is the practice of protecting critical systems and sensitive information from digital attacks"* (IBM, s. f., párr. 1). Esto incluye desde la seguridad de redes y dispositivos, hasta la gestión de identidades y el resguardo de aplicaciones y servicios en la nube. Así, la ciberseguridad se concibe como un enfoque integral que involucra tecnología, procesos y a las personas que interactúan con los sistemas.

En cuanto a los tipos de ciberataques más comunes, se pueden identificar los siguientes:

- *Malware*: software malicioso diseñado para dañar sistemas o robar información.
- *Ransomware*: bloquea el acceso a los sistemas y exige un rescate económico para devolverlo.
- *Phishing*: engaños a través de correos electrónicos o mensajes para obtener datos sensibles.
- Robo de credenciales: ataques dirigidos a obtener nombres de usuario y contraseñas.
- *DDoS* (Denegación de servicio distribuido): sobrecarga de servidores para interrumpir servicios.
- Amenazas internas (Insider threats): riesgos originados por empleados o personas con acceso autorizado.
- *Cryptojacking*: uso ilícito de dispositivos para minar criptomonedas.
- Ataques basados en IA: técnicas que aprovechan la inteligencia artificial para hacer más efectivos los ataques.

"Cyberattacks are becoming more sophisticated, and attackers are using new techniques to gain access to systems, steal data and disrupt operations" (*IBM*, s. f., párr. 4). Esto refleja la necesidad de que la ciberseguridad evolucione constantemente, fortaleciendo tanto las herramientas tecnológicas como la capacitación de los usuarios.

2.2 Factores humanos

Los factores humanos corresponden a características del comportamiento humano dados diversos contextos. En el caso de la ciberseguridad, estos se refieren al rol que adquiere el comportamiento y las decisiones de las personas en la seguridad de los sistemas de información. Como señala un estudio, *"the role of human behaviour in coping with cyberattacks and strengthening cyber defences is grouped into the theme of 'human factors' in cybersecurity"* (*Nifakos* et al., 2021). Los ciberdelincuentes están explotando cada vez más las vulnerabilidades humanas a través de técnicas como la ingeniería social, en lugar de centrarse únicamente en fallas técnicas.

"In complement to cyberattacks, which have been targeted against the vulnerabilities of information technology (IT) infrastructures, a new form of cyber attack aims to exploit human vulnerabilities; such attacks are categorised as social engineering attacks" (*Nifakos* et al., 2021). Los ataques de ingeniería social, como el phishing, son un ejemplo claro de cómo se busca manipular a los individuos para que divulguen información sensible o realicen acciones que comprometan la seguridad.

Un concepto fundamental para contrarrestar estos riesgos es la ciberhigiene. Concepto que se desarrollará en el siguiente subcapítulo.

Adicionalmente, el estudio de *Kalhoro* et al. (2021), identifica cinco categorías principales de factores que influyen en el comportamiento de los usuarios en ciberseguridad:

1. **Factores personales**: Aquellos relacionados con las características individuales que tienen una influencia significativa en el comportamiento del usuario.
2. **Factores sociales**: Relacionados con la influencia de otros individuos y la comunidad en el comportamiento de un usuario.
3. **Factores sociocognitivos**: Referidos a los procesos mentales y creencias que guían las acciones de los usuarios.
4. **Factores ambientales**: Aspectos del entorno físico y de trabajo que pueden afectar la seguridad.
5. **Factores tecnológicos**: Características de los sistemas y tecnologías que influyen en cómo los usuarios interactúan con ellos.

El proceso de investigación para cada factor, se puede dividir en tres fases principales:

- **Planificación**: Proceso en el que se definen las preguntas de la investigación, la estrategia de búsqueda, las palabras clave y las fuentes de datos.
- **Conducción**: Incluye la búsqueda y selección de estudios primarios, el establecimiento de criterios de inclusión y exclusión, la evaluación de la calidad de los estudios y la extracción de los datos necesarios.
- **Reporte**: Finalmente, se sintetizan los datos obtenidos, se presentan los resultados y se elaboran las conclusiones y la discusión.

Esto puede ser relevante para comprender los distintos criterios y procedimientos que serán la base del desarrollo de las siguientes secciones del presente trabajo.

2.3 Ciberhigiene

La ciberhigiene es un concepto relativamente reciente cuya definición ha sido explorada por varios autores a lo largo de los años, en general era entendida como el conjunto de prácticas rutinarias que buscan proteger la integridad de dispositivos, redes y datos en el ciberespacio. Esto con la finalidad de que los usuarios logren reducir brechas de seguridad y evitar vulnerabilidades.

Estas fueron las principales definiciones exploradas:

- *Husain Aloul (2012)*: Define la ciberhigiene en términos de conocimientos y comportamientos que los usuarios deben mantener respecto al uso de software de seguridad, autenticación, prevención del phishing, cuidado en redes sociales, navegación web segura, así como manejo de redes Wi-Fi y dispositivos USB.
- *Lawrence Laws (2015)*: La concibe como un conjunto de prácticas en Internet orientadas a proteger tanto los dispositivos como la información personal de los usuarios.
- *Kalhoro, Abbasi, Memon y Khuwaja (2021)*: Asocian la ciberhigiene con comportamientos rutinarios, tales como verificar la presencia de virus, utilizar contraseñas seguras y mantener actualizado el software.
- *David Kirkpatrick (2016)*: La relaciona con la implementación y cumplimiento de políticas de seguridad de datos, buscando minimizar daños y reducir la superficie de ataque dentro de las organizaciones.
- *Farwell y Rohozinski (2011)*: Destacan la importancia de formar una fuerza laboral educada, capaz de reducir errores humanos que puedan conducir a intrusiones cibernéticas.
- *D. Dodge (2014)*: Resalta el papel fundamental de cada empleado que utiliza un computador dentro de la organización, enfatizando la responsabilidad individual en la seguridad digital.

Más allá de estas definiciones, resulta especialmente relevante la propuesta de Vishwanath, quien conceptualiza la ciberhigiene como *"a set of cybersecurity practices that online users should adopt to safeguard the security and integrity of their personal information on internet-connected devices against potential cyber threats"* (*Vishwanath* et al., 2020, p. 2). Esta definición destaca porque no se limita al ámbito técnico, sino que integra dimensiones conductuales, culturales y contextuales. Además, Vishwanath sostiene que la ciberhigiene debe entenderse como un conjunto de prácticas flexibles y escalables, adaptables al rol, la organización y el sector en el que se aplican, lo que la convierte en una noción más integral y empíricamente validada.

Un concepto central en la propuesta de Vishwanath es la importancia del *self-awareness* (auto-conciencia) por sobre el conocimiento puramente técnico. Según sus investigaciones, la eficacia de la ciberhigiene no depende tanto de que los usuarios comprendan el funcionamiento interno de una amenaza, sino de que sean capaces de reconocer los síntomas y señales de riesgo en su interacción cotidiana con la tecnología, similar a como ocurre con una enfermedad en el área de salud. Con base en esta perspectiva, la ciberhigiene se concibe como multidimensional, abarcando distintos ámbitos del comportamiento digital. Estas dimensiones pueden agruparse de la siguiente manera:

- **Higiene del almacenamiento y dispositivos**: actualizar software, instalar parches de seguridad, emplear antivirus y evitar el uso de memorias *USB* no confiables.
- **Higiene de la transmisión de información**: garantizar la transmisión segura mediante cifrado, certificados digitales y redes privadas virtuales (*VPN*).
- **Higiene en redes sociales**: gestionar adecuadamente la privacidad, limitar la exposición de datos, supervisar contactos y prevenir fugas de información.
- **Higiene de la autenticación y credenciales**: crear contraseñas fuertes y únicas, utilizar autenticación multifactor y proteger las credenciales mediante cifrado.
- **Higiene en el correo electrónico y mensajería**: verificar la autenticidad de los mensajes, identificar fraudes o intentos de phishing y evitar caer en ataques de ingeniería social.

La investigación de *Vishwanath* et al. (2020) consolidó este enfoque mediante la creación del *Cyber Hygiene Inventory* (CHI), compuesto por 18 ítems distribuidos en cinco dimensiones (*Safety, Authentication, Firewall, Transmission y Yourself*). Esta herramienta permite medir el grado de ciberhigiene de los usuarios, mostrando que mientras mayor es la conciencia (*awareness*), más seguras son sus prácticas digitales.

En conclusión, la ciberhigiene corresponde a las prácticas de seguridad cibernética que los consumidores en línea deben seguir para proteger la seguridad e integridad de su información personal en dispositivos habilitados para internet. Estas prácticas deben ser flexibles y escalables según el rol, el tipo de organización y el sector en que se aplica.

2.4 Modelos de lenguaje

Los Modelos de Lenguaje Grande (*LLM*, por sus siglas en inglés) son sistemas de inteligencia artificial entrenados con vastas cantidades de datos textuales, capaces de comprender y generar lenguaje humano de manera coherente y contextualmente relevante. *OpenAI GPT-5*, *Gemini Flash 2.5*, entre otros, corresponden a *LLMs*, cuya principal característica es poseer datos masivos y más de 10 billones de parámetros, por lo que su propósito es ser capaces de realizar tareas complejas, como traducción automática, generación de texto y análisis de sentimientos, sin necesidad de información adicional específica para cada tarea.

Sin embargo, los *LLM* presentan desafíos en términos de consumo de recursos computacionales y costos operativos. Además, su tamaño y complejidad pueden dificultar su implementación en dispositivos con limitaciones de hardware o en aplicaciones que requieren respuestas rápidas y eficientes, por lo que implementar estas herramientas en un entorno local sin conexión a Internet no suele ser rentable ni viable sin un gran conjunto de máquinas interconectadas y con carga distribuida entre sí.

Es por esto, que los Modelos de Lenguaje Pequeño (*SLM*, por sus siglas en inglés) emergen como una alternativa eficiente. Estos modelos están diseñados para tareas específicas y son significativamente más pequeños en tamaño, lo que les permite operar con menor consumo de recursos y mayor velocidad. Según Belcak et al. (2025), *"Small language models are sufficiently powerful, inherently more suitable, and necessarily more economical for many invocations in agentic systems, and therefore are the future of agentic AI"* (Belcak et al., 2025, arxiv.org).

Los *SLM* son particularmente útiles en aplicaciones donde se requiere eficiencia y especialización, como en sistemas de recomendación, asistentes virtuales y análisis de datos en tiempo real. Esto los hace especialmente atractivos para el desarrollo del presente trabajo, ya que su capacidad para adaptarse a contextos específicos y operar en entornos con recursos limitados los convierte en opciones mucho más viables y flexibles que los *LLMs* más utilizados en el mercado.

2.5 RAG

La Generación Aumentada por Recuperación *(Retrieval-Augmented Generation, RAG)* es una técnica que combina la capacidad de generación de texto de los modelos de lenguaje con la recuperación de información externa en tiempo real. Esto permite a los sistemas generar respuestas más precisas y contextualmente relevantes al acceder a fuentes de información actualizadas y específicas.

Componentes Clave de *RAG*:

- *JSON Schema*: Esquema estructurado que define la organización y validación de los datos, facilitando la interoperabilidad y consistencia en el manejo de la información.
- *Chunking*: Proceso de dividir grandes volúmenes de texto en fragmentos más pequeños y manejables, lo que mejora la eficiencia en la recuperación y generación de información.
- *Base de Conocimiento (KB)*: Repositorio estructurado de información que sirve como referencia central para un sistema, organizando datos y conocimientos relevantes de manera accesible y consultable.
- *Snippet*: Fragmento individual de información o conocimiento extraído o generado, que contiene datos, recomendaciones o contenido específico, y que se utiliza como unidad básica de recuperación dentro de un sistema *RAG*.
- Aplicación en Ciberhigiene: *RAG* permite integrar información actualizada y específica del contexto del usuario, proporcionando recomendaciones personalizadas y adaptativas en tiempo real. Esto es particularmente útil en el ámbito de la ciberhigiene, donde las amenazas y mejores prácticas evolucionan constantemente. La *KB* sería la colección de información, y los snippets los fragmentos de texto con recomendaciones, buenas prácticas, ejemplos o riesgos específicos. El *RAG* consulta estos snippets para generar recomendaciones precisas y adaptadas a cada usuario.

2.6 IMECH

El Instrumento de Medición de Ciberhigiene (IMECH) en su versión 0.2 fue desarrollado por el Dr. Claudio Alvarez y su equipo, basándose en el modelo de ciberhigiene de cinco factores de Vishwanath et al. (2020). Este instrumento está diseñado para evaluar la ciberhigiene en instituciones públicas o en aquellas que manejan información crítica, como hospitales y puertos.

La evaluación se realiza a través de las siguientes dimensiones, visibles en la figura 2.1:

- **Higiene de Dispositivos y Almacenamiento de Información (DAI)**: Esta dimensión se refiere a las prácticas cotidianas para resguardar la integridad física y digital de los dispositivos utilizados en el entorno laboral. Esto incluye teléfonos móviles personales y computadoras de escritorio o portátiles. Cubre la seguridad de la información almacenada en los dispositivos o transferida entre ellos. También incluye conductas como la

actualización periódica de software, el uso de mecanismos de bloqueo, la instalación responsable de aplicaciones, el respaldo de información y la separación de cuentas personales y laborales.

- **Higiene de la Transmisión de Información (TRI)**: Se enfoca en las prácticas para minimizar el riesgo de exposición o intercepción de información sensible durante su circulación. Incluye el uso de canales seguros para compartir datos, evitar el envío de contraseñas o datos sensibles por mensajería o correo electrónico, y la eliminación oportuna de mensajes y archivos confidenciales. También evalúa la conciencia sobre los riesgos de usar redes Wi-Fi públicas y medios de almacenamiento extraíbles como pendrives.
- **Higiene del Comportamiento en las Redes Sociales (CRS)**: Esta dimensión se refiere a las prácticas conscientes que adoptan las personas en plataformas sociales para proteger su privacidad y resguardar la información personal y profesional. Contempla la forma en que se comparte información propia o ajena, el grado de protección configurado en los perfiles y la actitud frente a la veracidad del contenido.
- **Higiene de la Autenticación y el Uso de Credenciales (AUC)**: Se relaciona con las prácticas, actitudes y percepciones sobre cómo las personas crean, gestionan y protegen sus contraseñas y credenciales de acceso digital. Incluye comportamientos relacionados con la complejidad, unicidad y periodicidad de cambio de las contraseñas, así como la conciencia sobre los riesgos de reutilizar claves simples.
- **Higiene del Correo Electrónico y la Mensajería (MCE)**: Esta dimensión se refiere a las prácticas y disposiciones cognitivas para identificar, prevenir y evitar comportamientos de riesgo en la comunicación digital por correo y mensajería. Incluye la capacidad de reconocer mensajes potencialmente maliciosos, la prudencia frente a enlaces y archivos adjuntos, y la disposición a verificar solicitudes inusuales. También abarca el uso de cuentas institucionales y la separación de correos personales y laborales.

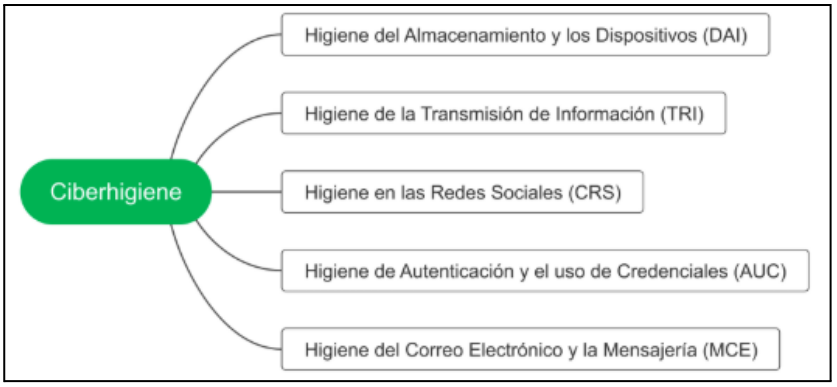

Figura 2.1 - Modelo de Cinco Factores de Ciberhigiene de *Vishwanath* et al., 2020
Fuente: Reporte HLT 2024-25

Adicionalmente, el *IMECH* también incluye preguntas para evaluar la alfabetización digital en ciberseguridad, la cual se refiere al grado de familiaridad del personal con conceptos fundamentales sobre seguridad informática. Las dimensiones de conocimiento que se evalúan son:

1. Conocimiento sobre sistemas de autentificación.
2. Conocimiento sobre sistemas de comunicación segura.

3. Conocimiento sobre amenazas basadas en la ingeniería social.
4. Conocimiento sobre riesgos de seguridad.
5. Conocimiento sobre el comportamiento adecuado.

Con lo anterior, el *IMECH v0.2* se presenta como una herramienta multidimensional y flexible, que permite obtener una visión exhaustiva de los hábitos de ciberhigiene y del nivel de conciencia del usuario frente a los riesgos cibernéticos, facilitando la identificación de áreas de mejora y la presentación de la información recopilada.

Esta información podrá emplearse para generar reportes informativos, que a su vez podrían servir como base para acciones de capacitación o prevención dentro de las organizaciones. En el contexto del presente trabajo, el *IMECH* se utilizará para elaborar reportes individuales personalizados para cada usuario de la organización donde se aplicó el instrumento.

2.7 HFShield

"HFShield es una plataforma desarrollada con el objetivo de gestionar de manera integral los factores humanos en ciberseguridad" (Avendaño Palacios, 2025, p. 1), enfocándose en la evaluación y seguimiento de hábitos de ciberhigiene de los usuarios dentro de instituciones que manejan información sensible.

El sistema HFShield se estructura en tres aplicaciones interconectadas que interactúan mediante una API REST segura, visible por completo en la figura 2.2:

- ***Sentinel**:* Es el subsistema público de cara al usuario. Permite leer y firmar el consentimiento informado, acceder al instrumento de medición y recuperar los reportes individuales con retroalimentación personalizada. El frontend de Sentinel está desarrollado en *ReactJS* y se comunica con *Command* a través de solicitudes *HTTP* a la *API* correspondiente.
- ***Command**:* Es la aplicación de administración, accesible únicamente desde direcciones *IP* autorizadas. Permite crear, visualizar, editar y eliminar proyectos, así como asociar los instrumentos de medición. *Command* cuenta con autenticación de doble factor mediante *WebAuthn* y se desarrolló usando *Ruby on Rail*s para el backend y *ReactJS* para el frontend. Administra los datos a través de una base de datos *PostgreSQL*, asegurando la integridad y seguridad de la información.
- ***Relay**:* Se encarga de la generación de reportes, tanto institucionales como individuales, en formato *PDF*. Utiliza *Python* como lenguaje central, integrando librerías como pandas y *Jinja2*, gráficos generados en *R* mediante *ggplot2*, y la generación de documentos estructurados con *LaTeX*. La generación de reportes se automatiza y personaliza mediante el uso de *GPT-4*, accediendo a los datos de medición a través de la *API* de *Google Sheets* y comunicándose con *Command* mediante una *API REST* segura.

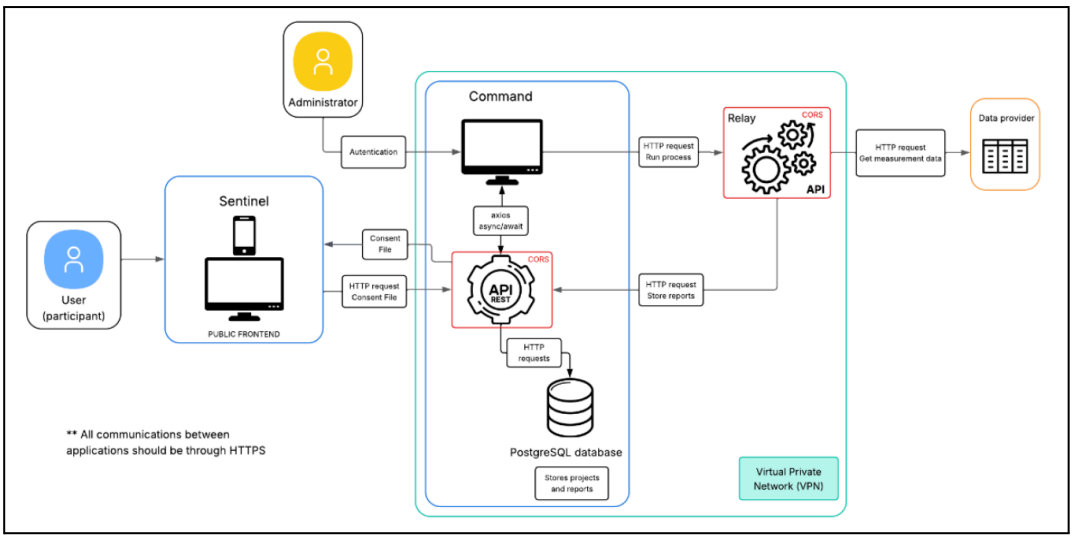

Figura 2.2 - Diagrama físico de la arquitectura *HFShield*

Fuente: HFShield: Plataforma para la gestión integral de factores humanos en ciberseguridad

La arquitectura de HFShield sigue un enfoque *secure by design*, incorporando medidas de seguridad como autenticación robusta, cifrado de datos y archivos, control de acceso a funcionalidades críticas, y comunicación segura entre los subsistemas mediante *mTLS* sobre redes privadas (*VPN*). Este diseño permite minimizar la exposición de información sensible, garantizar la integridad de los datos y asegurar la escalabilidad del sistema. Los contenedores y servicios principales se observan en la figura 2.3.

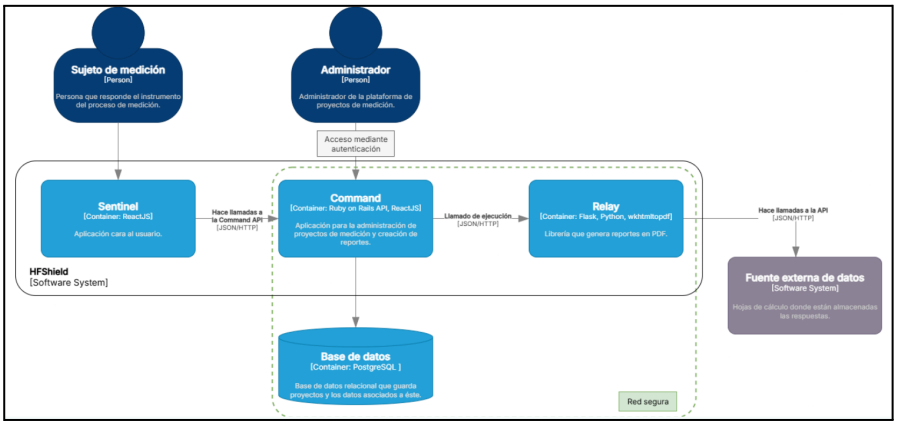

Figura 2.3 - Diagrama de componentes C4

Fuente: HFShield: Plataforma para la gestión integral de factores humanos en ciberseguridad

3. Enfoques arquitectónicos

En esta sección se exponen las alternativas de enfoques arquitectónicos más apropiados para cumplir el objetivo del presente trabajo. Esto en una vista de alto nivel e intentando hallar el flujo más eficiente y conveniente para el proceso.

3.1 El proyecto a gran escala

El proyecto surge de la necesidad de reforzar la ciberhigiene en instituciones públicas que carecen de personal capacitado para proteger adecuadamente la información, especialmente en sectores críticos como hospitales y puertos. Según el INCIBE (2024), el sector sanitario es uno de los más atacados a nivel mundial debido al alto valor de sus datos y su creciente conectividad, siendo el ransomware, el robo de datos y las intrusiones no autorizadas los incidentes más comunes, generalmente motivados por fines económicos y crimen organizado.

Frente a esta situación, el equipo del Dr. Claudio Alvárez, desarrolla la plataforma HF Shield, que propone evaluar y fortalecer la ciberhigiene mediante la aplicación del IMECH, un instrumento basado en el modelo de cinco dimensiones de Vishwanath et al. (2020). Este instrumento mide prácticas y conocimientos en torno a la seguridad digital del personal, proporcionando una visión detallada de fortalezas y debilidades.

A partir de esta información, el proyecto busca entregar reportes organizacionales que orienten decisiones institucionales. Sin embargo, dado que la ciberhigiene depende de cada usuario dentro de la organización, también debe desarrollar reportes individuales con recomendaciones personalizadas para cada encuestado. Con ello, se abre la posibilidad de utilizar los resultados no solo como diagnóstico, sino también como insumo para futuras capacitaciones, programas de prevención y otras acciones específicas de mejora en ciberseguridad.

3.2 El desafío del proyecto y la solución propuesta

El desafío central de este trabajo es diseñar un mecanismo para que cada encuestado reciba un reporte individual de ciberseguridad con recomendaciones personalizadas, de forma clara y accesible. El problema no es solo medir la ciberhigiene con el IMECH, sino también hacer llegar la información de los resultados al usuario final, de modo que pueda comprenderla y ponerla en práctica.

Para ello, la solución propuesta consiste en un proceso de múltiples fases donde los resultados del IMECH (respuestas Likert, porcentajes y promedios) se transforman en informes cualitativos con recomendaciones adaptadas. Esto se busca implementar utilizando modelos de lenguaje (LLM o SLM) como “narradores”, encargados de elaborar tanto las recomendaciones como el texto de cada reporte.

El desafío técnico de la solución consiste en procesar de manera masiva los resultados de numerosos encuestados y producir sus informes de forma segura y automatizada, garantizando coherencia entre los datos de entrada y las recomendaciones presentadas.

Los reportes, de una a dos planas de extensión, deben estar respaldados en fuentes confiables y redactados en un lenguaje claro y no técnico. Su objetivo es brindar al usuario orientaciones prácticas, inspiradas en el concepto de una “receta médica”: así como un médico prescribe medicamentos para tratar o prevenir una enfermedad, el informe de ciberhigiene entrega recomendaciones precisas que permitan fortalecer la seguridad digital, corregir deficiencias e incluso reducir brechas críticas cuando estas están presentes.

3.3 Alternativas de enfoques arquitectónicos

El módulo de generación de reportes individuales dentro de *Relay* debe integrarse de manera coherente a la arquitectura global de *HFShield* (ver sección 2.7), respetando la filosofía *secure by design* y la restricción de que Relay solo se comunica con Command a través de una *API REST* protegida en la *VPN* interna. De este modo, *Relay* actúa como un servicio cerrado, expuesto únicamente a solicitudes validadas desde Command, tal como se observa en la figura 3.1.

Figura 3.1 - Vista en alto nivel de HF Shield
Fuente: Elaboración propia

El objetivo principal es transformar los resultados del *IMECH* en reportes individuales claros, comprensibles y confiables, apoyándose en modelos de lenguaje y, en algunas alternativas, en una base de conocimiento. A nivel conceptual, se identifican tres enfoques posibles para generar los informes dentro de *Relay* (módulo de reportes individuales):

1. *LLM* directo:
   Los resultados del *IMECH* se envían directamente a un modelo de lenguaje grande (*LLM*), que genera el reporte sin apoyo de una base de conocimiento adicional. Esto se aprecia en la figura 3.2.

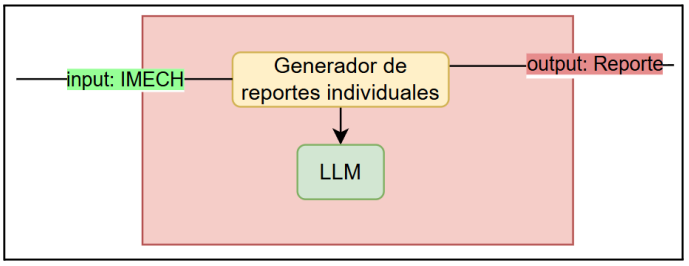

Figura 3.2 - Opción 1 de enfoque arquitectónico dentro de *Relay*
Fuente: Elaboración propia.

Ventajas: Alta flexibilidad, personalización inmediata y simple implementación..
Desventajas: Elevado costo por uso del servicio, riesgo de alucinaciones y menor trazabilidad de las recomendaciones.

2. Plantillas predefinidas:
   Se emplean bloques de texto y reglas asociadas a rangos de puntajes de la encuesta (por ejemplo, 3 de 5 puntos en un ítem, se aplica una recomendación concreta dado ese ítem y puntaje). La arquitectura de esta alternativa se aprecia en la figura 3.3.

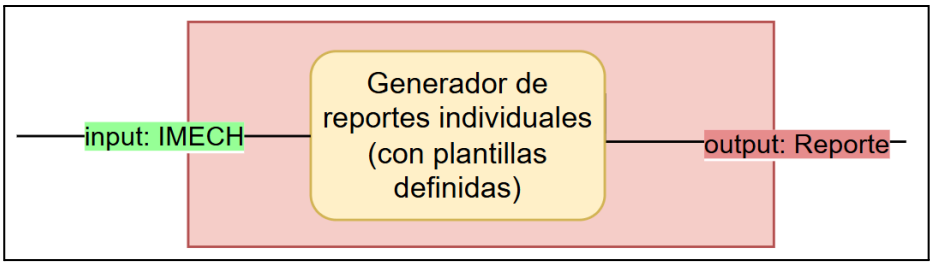

Figura 3.3 - Opción 2 de enfoque arquitectónico dentro de *Relay*
Fuente: Elaboración propia

Ventajas: Muy bajo costo, rápida generación de informes.
Desventajas: Informes repetitivos y genéricos, con narrativa limitada.

3. *KB* + narrador:
   Se utiliza una *KB* con *snippets* curados y estructurados como base de conocimiento, y un modelo de lenguaje actúa como narrador para generar los reportes individuales. Esta alternativa se aprecia en la figura 3.4.

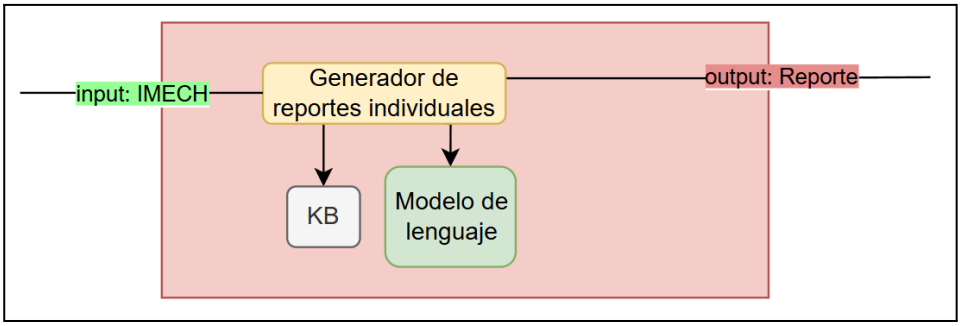

Figura 3.4 - Opción 3 de enfoque arquitectónico dentro de *Relay*
Fuente: Elaboración propia

Ventajas: Personalización, coherencia y trazabilidad, manteniendo el contenido basado en fuentes verificadas.
Desventajas: Requiere preparación inicial de la KB y manejo de dos flujos asíncronos, es más complejo de implementar.

3.4 Enfoque arquitectónico propuesto

A partir de lo explorado en la anterior sección, la opción 3 (*KB* + narrador*)* es la más completa y precisa, con lo que el enfoque arquitectónico propuesto requiere organizar:

- Los resultados del *IMECH* enviados desde *Command*.
- La interacción con un modelo de lenguaje (*LLM/SLM*).

- El uso de una base de conocimiento enriquecida con *RAG*.
- El resto de módulos dentro de *Relay* para contextualizar el origen de la solución.

Con esto, la figura 3.5 contempla todos estos requisitos y representa el enfoque arquitectónico que se planea desarrollar para el proyecto.

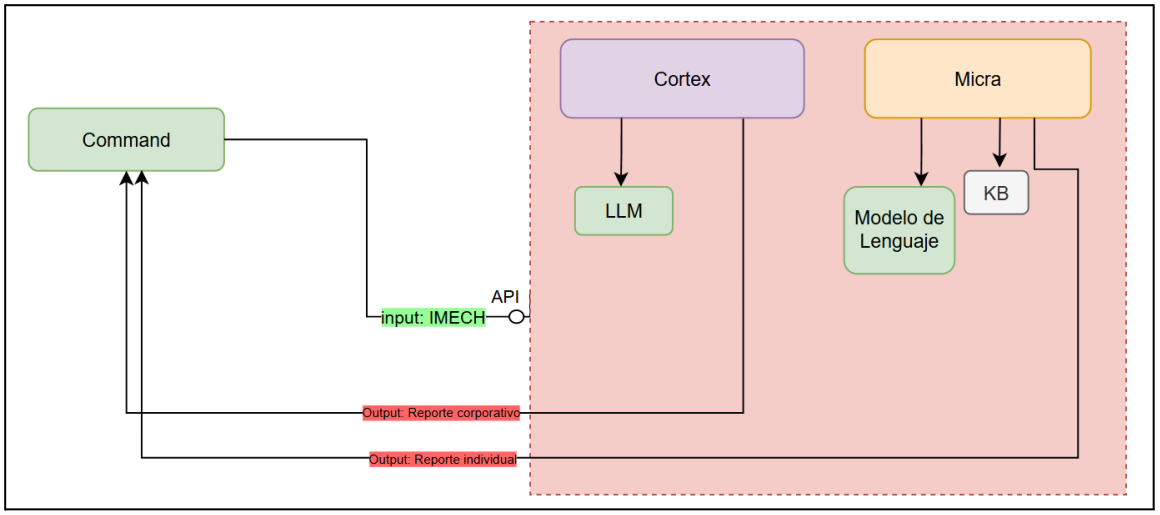

Figura 3.5 - Enfoque arquitectónico propuesto para *Relay*
Fuente: Elaboración propia

El módulo *Cortex* es el encargado de la generación de reportes generales con un enfoque corporativo. Si bien no forma parte del proyecto tratado en este documento, también constituye un componente dentro de *Relay*.

Por su parte, *Micra* se encarga de generar reportes individuales a partir de los resultados del *IMECH*. Este módulo contendrá un orquestador (Se comentará más al respecto en la sección 5) para manejar las interacciones y comunicación entre el modelo de lenguaje y la base de conocimiento (*KB*), la cual funciona como “memoria central” para todos los informes.

Ambos módulos, dentro de *Relay*, se comunican con *Command* a través de una *API*. De esta manera, *Command* puede solicitar a *Relay* la ejecución de los trabajos de cada módulo de forma independiente, obteniendo así los reportes corporativos o individuales según corresponda y permitiendo que puedan ser consultados por el usuario vía *Sentinel*. Esto asegura que *Relay* se mantenga aislado de *Command*, con la misión de obtener reportes coherentes, comprensibles y seguros.

4. Comparación de modelos de lenguaje

En la presente sección se comparan las posibles alternativas de modelos de lenguaje disponibles para el desarrollo del proyecto en *Micra* (*Relay*).

4.1 Rol del modelo de lenguaje

Antes de seleccionar un modelo de lenguaje concreto, es imprescindible definir claramente para qué se empleará dentro del flujo de *Relay* y qué tareas debe cumplir, pues a partir de este análisis, el criterio de decisión sobre qué alternativa resulta más conveniente y eficiente en términos de calidad y costo, cambiará.

El modelo de lenguaje será la pieza encargada de transformar resultados cuantitativos (respuestas Likert y metadatos del *IMECH*) en reportes breves con datos cualitativos útiles para los usuarios finales, siguiendo el enfoque descrito en la Sección 3.2. Para este propósito, y basándose en el enfoque arquitectónico propuesto (figura 3.5), a continuación se describen las responsabilidades, requisitos y restricciones principales que debe afrontar el modelo de lenguaje en este proyecto.

1. **Generación de contenido legible y accionable**: Producir reportes individuales de una a dos planas en lenguaje claro, no técnico, que incluya un diagnóstico breve, recomendaciones concretas y prioridad de acciones (qué hacer primero, cómo hacerlo y por qué). Todo en el contexto de la ciberhigiene.
2. **Personalización**: Adaptar el texto al perfil del encuestado (nivel de conocimiento, rol en la institución, industria) para que las recomendaciones sean relevantes y aplicables.
3. **Soporte de trazabilidad**: Cada recomendación debe poder vincularse a las evidencias o snippets de la base de conocimiento (*KB*) que la respaldan (instituciones verificadas).
4. **Escalado y procesamiento masivo**: Permitir generación automática y segura de grandes volúmenes de reportes, preferentemente en modalidad batch con opciones de cola y paralelismo controlado.
5. **Privacidad y seguridad**: Evitar la exposición de datos sensibles. Esto será trabajo de la implementación final, pero es importante que el modelo de lenguaje no reciba información sensible innecesaria.
6. **Fiabilidad y control de alucinaciones**: El modelo debe ser lo suficientemente consistente y preciso para que todos los reportes generados mantengan el mismo formato y calidad.
7. **Explicabilidad y justificación**: A partir de la comunicación entre el modelo de lenguaje y la base de conocimiento, cada recomendación dada debe incluir una breve justificación y la referencia al ítem del *IMECH* evaluado.
8. **Control de estilo**: El modelo debe respetar un tono consistente (tipo “receta”, asertivo y práctico), con variables de salida (nivel de detalle, formalidad) a través de *Micra*.

Considerando estos 8 requisitos, en la siguiente sección se explorarán alternativas de LLMs (modelos grandes en la nube) y SLMs (modelos más pequeños y desplegables) que permitan cumplir adecuadamente el rol esperado del modelo de lenguaje para el presente proyecto.

Para más adelante puede ser necesario comprender la definición de “*token*”:

Token: Un token es la unidad mínima de texto que un modelo de lenguaje procesa; puede ser una palabra completa, parte de una palabra o incluso signos de puntuación y espacios. Los modelos no leen palabras directamente, sino estos tokens, y cada operación de entrada o salida consume tokens.Su importancia radica en que determinan el costo de usar un modelo en la nube, ya que muchas plataformas cobran por token, y también limitan la cantidad de contexto que el modelo puede manejar en un solo prompt, afectando la consistencia y calidad de la generación de texto.

4.2 Tabla comparativa LLMs

La tabla 4.1 presenta una comparativa de algunos modelos de lenguaje grandes como referencia inicial. Esta selección no pretende reducir la comparativa de LLMs a solo estas opciones, pues no son todos los modelos disponibles y compatibles con el rol descrito en la sección 4.1. Existen otros modelos de lenguaje, inclusive de los mismos desarrolladores, con costos mayores o menores y diferentes limitaciones de uso, que también podrían considerarse en etapas posteriores del proyecto, está solo es una primera aproximación a las alternativas más conocidas y de fácil acceso.

Modelo	**Autor**	**Acceso**	**Coste aprox. por 1000 tokens (en USD)**	**Costes/limitaciones adicionales**	**¿Puede cumplir los 8 requisitos?**
GPT-5 Nano	OpenAI	API	**~ $0,000585**	Coste por token muy bajo, **16K-32K tokens de contexto**, **no procesa documentos externos**, dependencia de API, **políticas de uso** y latencia de red, **límites de cuota** según el plan (el de pago son $25 USD).	Sí, sin problemas.
Gemini Flash-Lite	Google	API	**~ $0,00064**	Latencia en nube, posibles límites de solicitudes, coste operativo acumulado, **8K-32K tokens de contexto** y **no soporta archivos externos.**	Sí, aunque requiere RAG y supervisión para minimizar alucinaciones.
Mistral Tiny	Mistral AI	API o local	**~ $0,0034**	Puede ofrecer opciones self-hosted en algunos casos, costes de inferencia y cuotas vía nube. **4K-8K tokens de contexto** y **lectura de archivos mediante chunking. Dificultades con el español.**	Parcial, es muy eficiente pero puede ser inconsistente.

Claude Instant	Anthropic	API	**~ $0,0042**	Políticas de uso, límites de solicitudes, coste por uso más alto. **9K-100K tokens de contexto, documentos en bloques.**	Si.
GPT-3.5 Turbo	OpenAI	API	**~ $0,0039**	Latencia/cuotas; coste acumulado en generación masiva. Menos consistente para flujos complejos. **4K-16K tokens de contexto. Necesita chunking para textos extensos.**	Parcial, sirve para prototipos, pero para generación masiva puede presentar dificultades.
GPT-4o Mini	OpenAI	API	**$0.000195 (input) / $0.000780 (output)**	Soporte de ventana de contexto de **128K tokens de contexto**. Soporta **múltiples documentos en un solo prompt.**	Sí, con configuración adecuada
Vicuna-13B	LMSYS	API / Local	Coste por token solo si se usa servicio. **~ $0,013** en nube	Requiere hardware potente. **4K-16K tokens de contexto. Lectura de archivos mediante chunking externo.**	Puede cumplir la mayoría de requisitos, pero tiene alta latencia.
Deep Seek-R1	Deep Seek	API	**~ $0.0024**	Requiere GPU (similar a 7B). Descuento de hasta 75% en horas fuera de pico (16:30–00:30 GMT). **4K-16K tokens de contexto. No soporta la lectura de archivos. Dificultades con el español.**	Sí, está pensado para tareas de razonamiento.
LLaMA 2 13B	Meta	Local / API	Gratuito local. **~ $0.00075 (input) / ~ $0.001 (output)**	Requiere GPU potente (+12GB VRAM), latencia baja en local y límites de uso en nube según proveedor	Sí – con RAG y snippets, capaz de cumplir todos los requisitos
GPT-NeoX-20B	Eleuther AI	Local / API	Gratuito en local. En nube **~ $0.00004**	Requiere GPU muy potente (+24GB VRAM), latencia baja en local. **2K-20K tokens de contexto. No soporta la lectura de archivos. Español no tan natural.**	Sí, pero depende de los prompts.

Mistral 7B	Mistral	Local / API	Gratuito local. En nube **~ $0.000028 (input) / ~ $0.000054 (output)**	Requiere GPU de 8GB VRAM, limitado en tareas muy grandes. **4K-8K tokens. No soporta la lectura de archivos. Dificultades con el español.**	Sí, pero necesita buena coordinación con RAG.
Phi-3 small (7B)	Microsoft	Local / API	Sin coste.	Requiere +12GB VRAM, tiene **licencia de uso**. **8K-16K tokens de contexto y soporte de archivos mediante chunking.**	Sí, es bastante potente.

Tabla 4.1 - Comparativa de LLMs

Fuente: Elaboración propia

4.3 Tabla comparativa SLMs

La tabla 4.2 muestra una comparativa de algunos modelos de lenguaje pequeños seleccionados como referencia inicial. Esta comparativa no abarca todas las alternativas disponibles ni implica que estas sean necesariamente las más adecuadas o convenientes para el proyecto. Existen muchos otros modelos con diferentes costos, requisitos o limitaciones de uso, que podrían evaluarse en etapas posteriores del proyecto. La presente selección se enfoca en algunas opciones destacables y las compara entre sí.

Modelo	Autor	Acceso	Coste aprox. por 1000 tokens (en USD)	Costes/limitaciones adicionales	¿Puede cumplir los 8 requisitos?
Phi-2 (2.7B)	Microsoft	Local	Sin coste.	Requiere GPU (12 GB VRAM según optimizaciones); mantenimiento y ajuste (LoRA/QLoRA). **4K-8K tokens de contexto y soporte de archivos mediante chunking. Requiere licencia de uso.**	Sí, con fine-tuning y RAG.

Nemotron-H (2-9B)	NVIDIA	Local / OEM	Sin coste en local. **~ $0.07 por 1M tokens en nube.**	Requiere GPUs con buen throughput; posibilidad de optimizaciones TensorRT; licencia/soporte comercial. **4K–16K tokens de contexto. No soporta la lectura de archivos.**	Sí, con optimizaciones.
Smol LM2 (0.125-1.7B)	Comunidad / Hugging Face	Local / HF Hub	Sin coste en local. **~ $0.096 por 1M tokens en nube.**	Menor VRAM; menos capacidad para outputs largos; puede requerir ensamblado de modelos múltiples. **2K-8K tokens de contexto, sin soporte para archivos. Casi nulo soporte para español.**	Parcial, puede referenciar datos de la KB pero su narrativa es limitada.
Hymba (1.5B)	NVIDIA (ej.)	Local	Sin coste.	Optimizada para *instruction following*; baja exigencia en VRAM, bajo soporte. **2K-4K tokens de contexto y lectura de archivos vía chunking. Casi nulo soporte para español.**	parcial, sirve para generación controlada.
Tiny Llama (1.1B)	Comunidad / Meta	Local / Hugging Face	~$0.096 por 1M tokens en nube.	VRAM modesta requerida; longitud limitada de contexto; menor fluidez en outputs largos. **2K-4K tokens de contexto y lectura de archivos vía chunking. Casi nulo soporte para español.**	Requiere buena configuración de plantilla y KB para evitar alucinaciones
GPT-2 Small (117M)	OpenAI	Local	Sin costo.	Output simple, limitaciones en lenguaje natural complejo; poca capacidad de razonamiento. **1K–2K tokens de contexto, sin soporte para archivos. Casi nulo soporte para español.**	Útil para partes del pipeline (traductor, resumen), no para narrativa rica completa

RETRO-7.5B	DeepMind-style (RAG hybrid)	Local + DB externa	Sin costo.	Requiere infraestructura para la DB externa (latencia de retrieval). **4K-16K tokens de contexto y lectura de archivos vía chunking. Bajo soporte para español.**	Sí, está especialmente diseñado para usar RAG.
xLAM-2 (8B)	Salesforce	Local / cloud	Sin costo.	Buen soporte para tool calling; 12GB de VRAM, políticas de uso, **4K-16K tokens de contexto y lectura de archivos vía chunking. Bajo soporte para español.**	Sí, aunque puede ser complejo de implementar.

Tabla 4.2 - Comparativa de SLMs
Fuente: Elaboración propia

4.4 Conclusiones respecto a los modelos de lenguajes

A partir de las tablas 4.1 y 4.2, se observó que modelos como *GPT-4o Mini, GPT-5 Nano* y *Claude Instant* presentan capacidades de contexto amplias, consistencia narrativa y costos por token suficientemente bajos como para permitir la generación masiva de reportes.

- *GPT-4o Mini* destaca por su ventana de contexto de 128K tokens y la posibilidad de procesar múltiples documentos en un solo prompt, lo que lo hace especialmente adecuado para la integración con la *KB* y para asegurar la trazabilidad de las recomendaciones.
- *GPT-5 Nano* mantiene un bajo coste operativo y una alta fidelidad en la generación de texto, aunque no soporta directamente la lectura de archivos externos.
- *Claude Instant*, pese a su mayor coste relativo y necesidad de supervisión para minimizar alucinaciones, ofrece estabilidad narrativa y tolerancia a textos extensos, lo que lo hace útil en contextos que requieran consistencia en reportes largos.

Por el contrario, *LLMs* como *Vicuna-13B* o *Mistral Tiny* presentan limitaciones importantes, ya sea en latencia, requisitos de hardware o consistencia, lo que llevó a descartarlos para la generación centralizada de narrativa.

En el caso de los SLMs, se constató que modelos como *LLaMA 2 13B, DeepSeek-R1* y *Phi-3 Small* podrían cumplir con la mayoría de los requisitos, siempre que se integren mecanismos de RAG y ajuste fino.

- *LLaMA 2 13B* permite trabajar en local o nube, con soporte para snippets de la KB y control sobre la generación de texto, aunque requiere *GPU* potente.
- *DeepSeek-R1* es eficiente y económico en nube para tareas de razonamiento, pero carece de soporte directo para lectura de archivos, limitando su autonomía en generación de reportes completos.

En cambio, SLMs más pequeños o con menor capacidad de contexto, como *SmolLM2, TinyLlama, Hymba* o *GPT-2 Small*, se mostraron insuficientes para generar narrativa completa de manera consistente y legible; por lo tanto, se consideraron inapropiados para ser usados como única fuente de generación de reportes. Por lo tanto, confiar en un *SLM* para la narrativa completa resultaría problemático debido a las limitaciones de contexto, consistencia y fluidez narrativa de la mayoría de estos modelos.
Sin embargo, los *SLMs* más robustos podrían desempeñar un papel complementario en un esquema híbrido, ya sea para pre-procesamiento de datos, validación de fragmentos de texto, generación de snippets o como mecanismo de fallback cuando se requiera mantener la información en local por motivos de privacidad. Con esto, de momento se utilizará uno de los *LLMs* mencionados para el desarrollo de *Micra*.

Aún así, el prácticamente nulo costo de los SLMs los hace muy llamativos para la solución final de este proyecto, pues si los reportes terminan siendo de unas 400 a 600 palabras con inputs *JSON*, con soporte en español e inglés y la capacidad de traducir recomendaciones provenientes del *RAG* (*KB*), finalmente el criterio de selección queda sujeto a los costos de la generación de los reportes en la práctica.
En la tabla 4.3 se presenta una comparativa final y aproximada de tokens por documento, lo que permite ver el desempeño teórico de los modelos más destacados en esta comparación.

Modelo	Autor	Acceso	Desempeño	Tokens máximos	Costo aprox por documento (USD)
GPT-5 Nano	OpenAI	API	Muy alto, consistente, soporte español/inglés	16K-32K	**$0,23-$0,92**
Claude Instant	Anthropic	API	Muy alto, soporte español/inglés	9K-100K	**$0,17-$1,88**
GPT-4o Mini	OpenAI	API	Muy alto, soporta múltiples documentos	128K	**$0,08-$0,31**
Deep Seek-R1	Deep Seek	API	Alto, pensado para razonamiento, soporte español/inglés	4K-16K	**$0,10-$0,40 (local ~$1000-$1500 ($0,30 a 4000 reportes); nube ~$18,96 por doc)**
LLaMA 2 13B	Meta	Local / API	Muy alto con RAG y snippets, soporte español	4K-16K	**~$0 local; inversión inicial ~$1500-$2500 GPU ($0,50 a 4000 reportes), 12-20s por doc**
Phi-3 Small (7B)	Microsoft	Local / API	Alto, soporte español, soporte de chunking	8K-16K	**~$0 local; inversión inicial ~$1500 GPU ($0,35 a 4000 reportes), ~15-25s por doc**

Tabla 4.3 - Comparativa final
Fuente: Elaboración propia

El análisis concluye que, una estrategia híbrida es la más adecuada: utilizar un LLM potente en nube para prototipos, supervisión y validación de resultados, mientras que los SLMs robustos locales se encargan de la generación masiva de reportes, preprocesamiento de datos, validación de fragmentos y traducción de recomendaciones. Esta combinación permite aprovechar la eficiencia y bajo costo operativo de los SLMs, asegurando soporte en español, integración con RAG y trazabilidad de recomendaciones. Con una expectativa de no superar los $0.5 USD por documento.

5. Modelo de procesos propuesto

En esta sección se propone un modelo de procesos para la generación de reportes individuales personalizados acorde a lo comentado hasta este punto del informe.

5.1 Análisis previo

En la sección 3 se exploró el enfoque arquitectónico a seguir para el desarrollo de *Micra*. Lo que resultó en la figura 3.5 con el enfoque arquitectónico propuesto, donde *Micra* contiene un orquestador que será responsable de recuperar los snippets desde la *KB*, construir el prompt y enviarlo al modelo de lenguaje. Con lo que el pre-procesamiento de los datos del *IMECH*, validación del informe generado y el envío de este reporte a *Command*, serían procesos internos en funciones de *Micra*.

Esto ya permite ilustrar el modelo de procesos en alto nivel de *Micra*, en particular, lo que se observa en la figura 5.1.

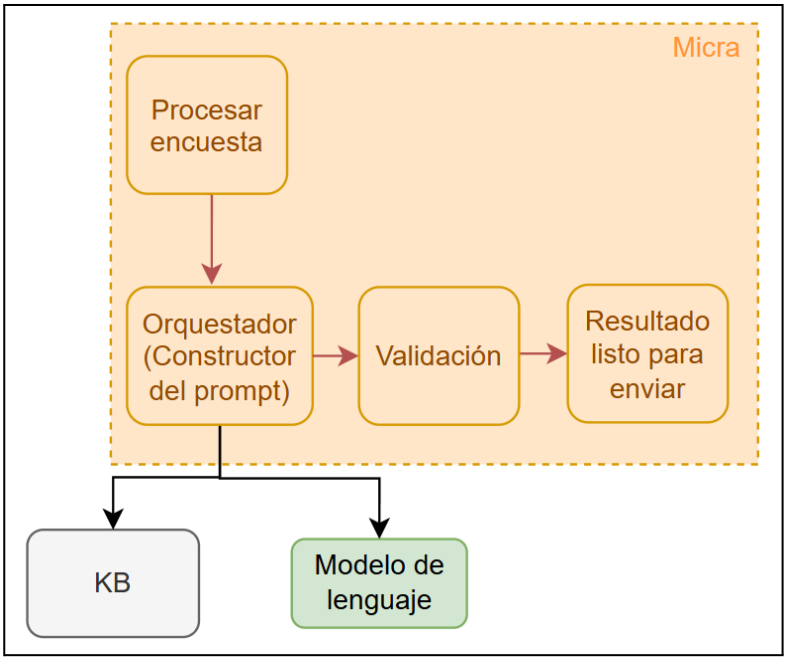

Figura 5.1 - Vista de procesos dentro de *Micra*
Fuente: Elaboración propia

5.2 Alternativas consideradas

A partir de la sección 5.1, lo que interesa evaluar ahora es analizar las distintas formas de componer los procesos dentro de *Micra* (procesos principales y orquestador), así como mostrar el flujo para la creación de la base de conocimiento (*KB*), pues esta requiere ser preparada y tratada antes de generar los reportes individuales.
De esta forma, lo que se propone es separar *Micra* y a la elaboración de la *KB* en dos flujos asíncronos, como se aprecia en la figura 5.2.

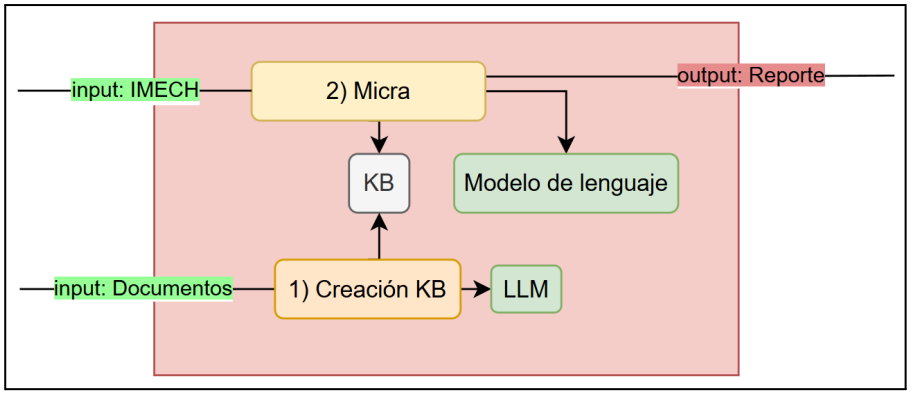

Figura 5.2 - Vista de alto nivel de los dos flujos en *Relay*
Fuente: Elaboración propia

1. **Flujo de creación y actualización de la base de conocimiento (*KB*)**

- Se encargaría de mantener una base de información estructurada y curada, que sirva como referencia para la generación de reportes.
- Este flujo se ejecutaría de forma independiente y asíncrona respecto a la generación de reportes, asegurando que los datos de referencia estén disponibles y actualizados cuando se necesiten.
- Los posibles enfoques arquitectónicos para este flujo serían:
    a. Orquestación centralizada con *LLM* grande: Emplear un *LLM* grande para llevar a cabo el flujo, encargado de procesar documentos confiables, generar *snippets* en formato *JSON*, y cargar los snippets en el *RAG* con parámetros de interés (atributos, roles de la industria, las 5 dimensiones, etc). Ver en figura 5.3.
    b. Microservicios especializados: Descomponer la ingestión de documentos, la extracción de *snippets* y la carga en *RAG* en módulos independientes, usando el *LLM* únicamente para la generación de recomendaciones. Ver en figura 5.4.

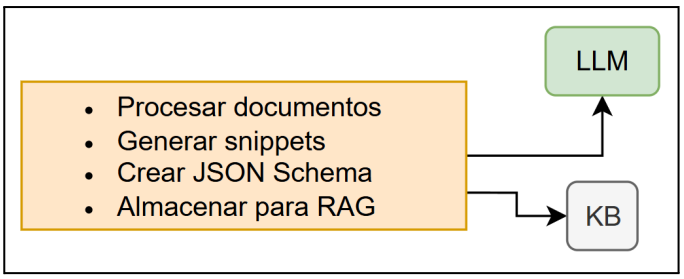

Figura 5.3 - Flujo 1 orquestación centralizada con *LLM* grande
Fuente: Elaboración propia

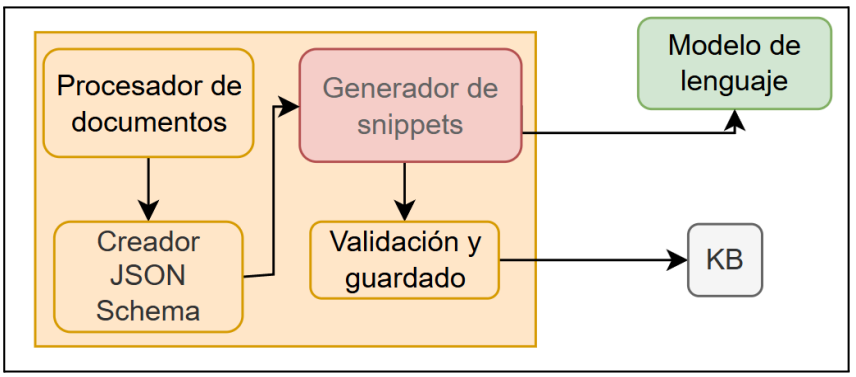

Figura 5.4 - Flujo 1 microservicios especializados
Fuente: Elaboración propia

2. Flujo de generación de reportes individuales (*Micra*)

- Este flujo se dispararía cada vez que llegan nuevos resultados del *IMECH* desde *Command*.
- Se apoyaría en la base de conocimiento (*KB*) y en modelos de lenguaje para producir informes personalizados y consistentes.
- Este flujo también podría retroalimentar la base de conocimiento, cerrando un ciclo conceptual de aprendizaje y mejora continua.
- Los posibles enfoques arquitectónicos para este flujo serían:
    a. Procesos centrados en *RAG*: Usar un *RAG* central como núcleo de recuperación de conocimiento. Luego usando un *SLM* como narrador de los informes. Ver en la figura 5.5.
    b. Orquestación híbrida *LLM offline* + *SLM* en línea: Un *LLM* grande que actualiza la *KB offline* y redacta el informe final, mientras que el *SLM* ligero apoya en el preprocesado y validaciones usando la base de conocimiento. Ver en la figura 5.6.
    c. Microservicios con validación automática: El proceso se descompone en módulos: procesamiento de encuestas, recuperación de snippets, generación narrativa, validación y renderizado. Y se comunican mediante *API*s internas seguras. Ver en figura 5.7.

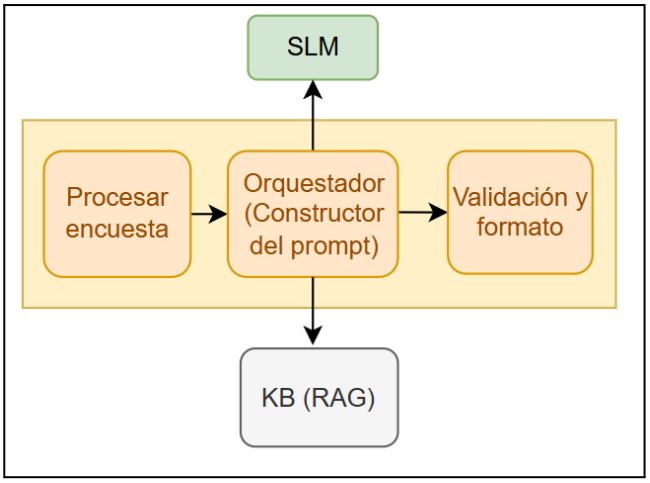

Figura 5.5 - Flujo 2 Procesos centrados en RAG
Fuente: Elaboración propia

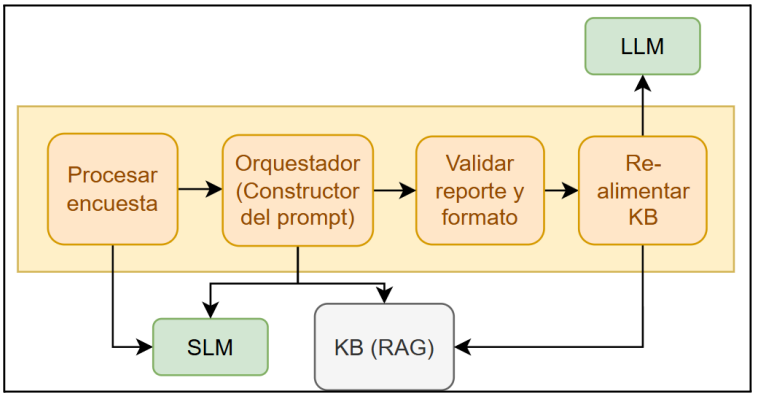

Figura 5.6 - Flujo 2 Orquestación híbrida LLM offline + SLM en línea
Fuente: Elaboración propia

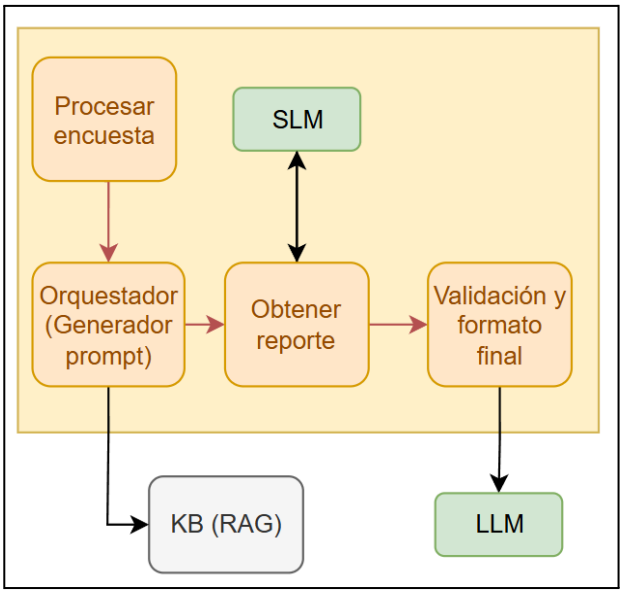

Figura 5.7 - Flujo 2 Microservicios con validación automática
Fuente: Elaboración propia

Finalmente, la figura 5.8 muestra los procesos principales que debe tener el orquestador dentro de *Micra*.

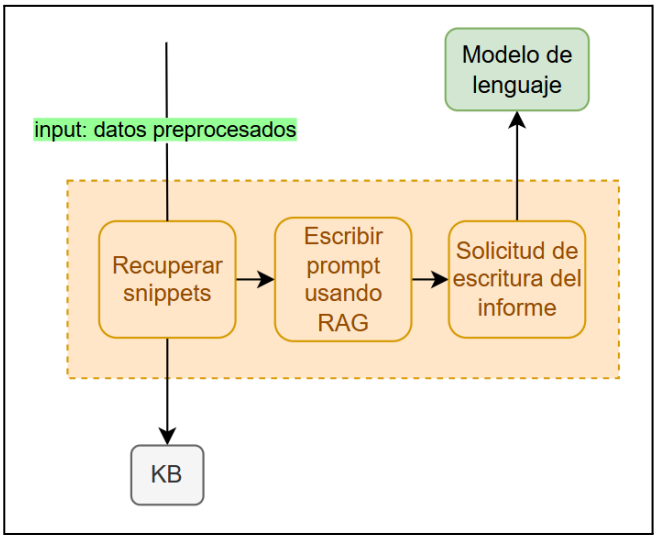

Figura 5.8 - Procesos dentro del orquestador
Fuente: Elaboración propia

5.3 Enfoque arquitectónico propuesto

A partir de las alternativas presentadas en la sección 5.2, se propone que el flujo de creación y actualización de la base de conocimiento (*KB*) utilice un *LLM* grande de manera offline para procesar documentos técnicos, generar snippets estructurados en formato *JSON* y cargarlos en el *RAG*. Este enfoque permitirá mantener una base de información curada y consistente de forma asíncrona, evitando sobrecargar la generación de reportes individuales, asegurando que los datos de referencia estén disponibles y actualizados cuando se necesiten, y facilitando la integración de nuevas fuentes o la expansión futura de la *KB*.

Por su parte, el flujo de generación de reportes individuales se apoyará en un *RAG* central junto con un *SLM* narrador, ejecutándose cada vez que lleguen nuevos resultados del *IMECH*.
Esta configuración garantizará que los informes sean precisos, coherentes y personalizados, optimizando recursos y escalabilidad, manteniendo la seguridad y el aislamiento de *Relay*, y permitiendo retroalimentar la *KB* para cerrar un ciclo de aprendizaje y mejora continua.

La combinación de ambos flujos, se puede visualizar en la figura 5.9 y se espera que proporcione un sistema eficiente, confiable y sostenible para generar reportes individuales de ciberhigiene dentro de *HFShield*.

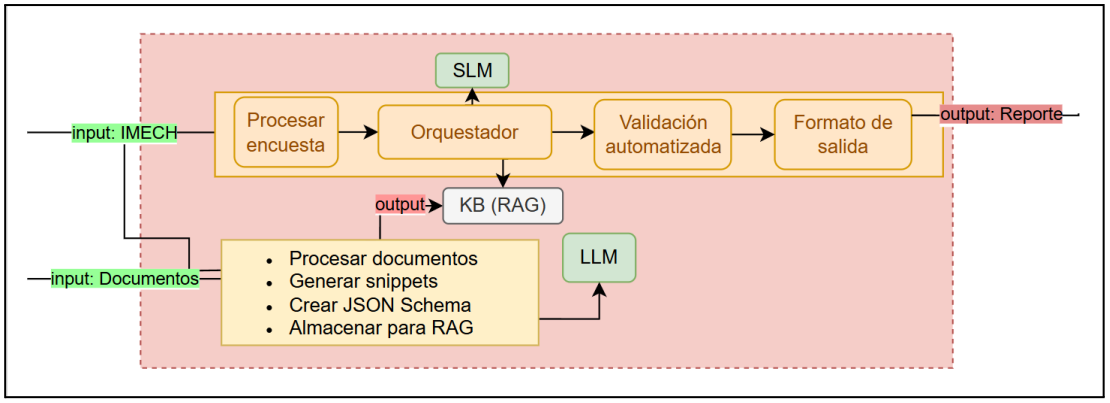

Figura 5.8 - Modelo de procesos propuesto
Fuente: Elaboración propia

5.4 Sobre la generación de la base de conocimiento y RAG

La base de conocimiento (*KB*) constituye el núcleo de referencia para la generación de reportes personalizados. Su creación implica un proceso estructurado de recopilación, preprocesamiento y transformación de documentos normativos, técnicos y de buenas prácticas, así como de resultados del *IMECH*, en fragmentos manejables (chunks). Estos fragmentos se convierten en snippets estructurados en JSON, incluyendo atributos como dimensión, rol, unidad, esfuerzo, impacto, pasos concretos y referencias verificables.

La generación de los snippets utiliza un *LLM* como "fábrica offline" para destilar los documentos en recomendaciones accionables, validando la consistencia, evitando la inclusión de PII y ajustando el contenido al contexto chileno. Cada snippet mantiene trazabilidad a su fuente, se valida automáticamente mediante JSON Schema y se revisa parcialmente mediante QA humano. Los snippets se almacenan en un *RAG* que permite a los modelos de lenguaje, generar reportes claros y personalizados, combinando datos cuantitativos con evidencia contextual. Este enfoque asegura consistencia, reproducibilidad y capacidad de actualización ante nuevas fuentes o cambios regulatorios, sin afectar la generación de reportes individuales.

Para lograr esto, el flujo de creación de la KB debe comenzar con la selección de los documentos que respaldan todas las recomendaciones y datos necesarios para la narración de los reportes. Se distinguen cuatro tipos principales:

1. IMECH: explica los ítems de la encuesta y sirve de referencia conceptual.
2. Documentos normativos y técnicos: proporcionan acciones y recomendaciones.
3. Documentos sobre amenazas y riesgos: describen consecuencias y riesgos asociados.
4. Documentos de comportamientos seguros: refuerzan buenas prácticas mediante ejemplos positivos.

Cada documento se preprocesa para obtener chunks (fragmentos manejables) de 200 a 400 palabras, manteniendo ideas completas. Luego se clasifican con un LLM mediante prompts claros, como el que se presenta a continuación.

“Eres un editor de contenido de ciberhigiene en salud. Con el pasaje dado, produce exclusivamente un JSON que cumpla el esquema provisto. No inventes políticas ni cifras. Redacta en es_CL, tono profesional y no culpabilizador. action debe ser imperativa. steps deben ser concretos y verificables. Incluye refs con source_uri#line_range.”

Los snippets generados contienen atributos definidos por una ontología mínima: dimensiones (DAI, TRI, CRS, AUC, MCE), roles (enfermería, médico, TI, administrativo), unidades (UCI, Urgencia, Pabellón, Imagenología, APS), esfuerzo (bajo, medio, alto), impacto (alto, medio, bajo), tags comunes (passwords, MFA, phishing, actualizaciones, backups, RDP, BYOD, EHR, correo) y locales (es_CL). Cada snippet incluye pasos concretos, esfuerzo/impacto y referencias verificables, evitando PII y expresiones culpabilizadoras.

Posteriormente, los snippets se validan automáticamente mediante JSON Schema y QA humano parcial. Los fragmentos se integran en un RAG usando técnicas de recuperación basadas similitudes entre vectores, lo que permite al SLM generar reportes personalizados a partir de los resultados de la encuesta. Los prompts para el LLM narrador podrían seguir un formato similar al siguiente ejemplo:

“Eres un generador de reportes de ciberhigiene en salud. Tu tarea es crear un informe individual basado únicamente en los datos de encuesta y los snippets proporcionados. No inventes cifras ni recomendaciones. Redacta en español chileno, tono profesional y no culpabilizador. Devuelve un JSON válido siguiendo el esquema: {dimension, hallazgo, riesgo, accion, ejemplo_positivo, effort, impact, referencias}.

Datos del usuario:
- Rol: Enfermería
- Unidad: UCI
- Resultados encuesta: {"AUC_3": 2, "TRI_1": 4, "DAI_2": 3}

Snippets recuperados (solo los más relevantes):
1. {"dimension":"AUC","text":"Nunca comparto mi contraseña de correo","action":"Configurar autenticación multifactor en todas las cuentas de correo","effort":"medio","impact":"alto","source":"NIST SP800-63B #4-10"}
2. {"dimension":"DAI","text":"Actualización de software no regular","action":"Habilitar actualizaciones automáticas en dispositivos médicos","effort":"bajo","impact":"alto","source":"ENISA Guidelines 2022 #12-14"}

Genera el reporte JSON final, máximo 300 palabras.”

Finalmente, la *KB* y el *RAG* permiten asociar hallazgos cuantitativos con recomendaciones prácticas y contextualizadas. Por ejemplo, un resultado bajo en un ítem de la dimensión AUC puede mapearse a un snippet normativo sobre manejo de contraseñas y un ejemplo positivo de conducta segura, generando un reporte final de 200–400 palabras en español chileno, listo para entrega o retroalimentación.

5.5 Consideraciones técnicas y desafíos futuros

Aunque el modelo de procesos propuesto establece un marco sólido, existen aún decisiones críticas que deberán abordarse en etapas posteriores del proyecto. Una de ellas corresponde al ciclo de vida de los reportes: debe definirse si estos serán generados bajo demanda, únicamente cuando el cliente los solicite, o si se almacenarán de manera persistente en un repositorio seguro. Cada enfoque conlleva implicancias distintas: la generación bajo demanda optimiza el uso de recursos y reduce riesgos de exposición, mientras que el almacenamiento facilita la trazabilidad histórica, pero exige mecanismos robustos de protección de datos.

En cuanto a la seguridad de la información, se deberá resolver cómo proteger datos sensibles como nombres, oficios o identificadores personales (incluido el RUT). El uso de anonimización, encriptación en reposo y en tránsito, así como controles de acceso basados en roles, se plantean como requisitos fundamentales.
Otro aspecto clave será definir el grado de acoplamiento entre los módulos de Micra, la KB y los modelos de lenguaje: un diseño demasiado rígido podría dificultar la integración futura de nuevas tecnologías, mientras que uno demasiado abierto podría comprometer eficiencia y control.

Finalmente, será necesario evaluar mecanismos adicionales de validación de calidad para los reportes finales, de modo que las recomendaciones sean consistentes, accionables y acordes al perfil de cada usuario. Estas consideraciones quedarán abiertas en esta etapa del trabajo, en tanto representan líneas de investigación y desarrollo que condicionarán la viabilidad y la solidez de Micra en un entorno real.

6. Esquema de datos propuesto

En esta sección se propone un esquema de datos para la *API* que se comunica con *Command*, con la finalidad de recibir la información necesaria para generar los reportes individuales en *Micra*.

6.1 Esquema de datos

A continuación se presenta el esquema de datos que la API recibiría desde *Command* para generar los reportes individuales. Este mismo esquema también es utilizado por *Cortex* en la elaboración de reportes corporativos dentro de su módulo. La diferencia entre ambos radica en la forma en que procesan la información para obtener los datos relevantes y presentar los reportes al usuario final.

[ESQUEMA DE DATOS]

Los aspectos específicos sobre qué datos de este *JSON schema* resultan de interés, dependerán de la implementación concreta del proyecto y de cómo se lleve a cabo la generación de reportes según la arquitectura y el modelo de procesos propuesto. Por lo que de momento, no se ahondará con mayor detalle en las decisiones futuras sobre los datos que efectivamente serán procesados y cuales serán ignorados.

7. Conclusiones

Tras la investigación bibliográfica, el análisis de alternativas arquitectónicas, la comparativa de modelos de lenguaje según costos e implementación, y proponer un modelo de procesos y un esquema de datos, se dispone de una base técnica y operativa sólida que permite avanzar a la fase de implementación del proyecto. El trabajo consolidó criterios claros sobre el propósito del sistema, los requisitos funcionales y no funcionales, y las restricciones de seguridad y trazabilidad que deben regir la generación de reportes individuales de ciberhigiene a partir del *IMECH*.

La decisión arquitectónica principal es construir *Micra* como módulo de *Relay*, expuesto a *Command* mediante una *API. Micra* recibe un *JSON Schema* con los resultados del *IMECH* y metadatos (Sección 6.1). Internamente, se operarán dos flujos asíncronos: uno encargado de la creación y actualización de la base de conocimiento (*KB*) mediante un *LLM offline* que procesa documentos verificados y genera *snippets JSON* validados; y otro dedicado a la generación masiva de reportes, que incluirá un orquestador encargado de recuperar snippets mediante *RAG* desde la *KB* y emplear un narrador para producir informes personalizados.
Esta separación permite mantener la *KB* como memoria central curada y reducir la carga y los riesgos en el flujo de generación de reportes.

En cuanto a la decisión de modelos de lenguaje, se propone un enfoque híbrido: usar *LLMs* en la nube para prototipado, orquestación y tareas de validación (primera etapa), y evaluar *SLMs* locales para procesos auxiliares o para la generación masiva cuando el rendimiento, el coste o la privacidad lo justifiquen (objetivo para el producto final). Este enfoque busca equilibrar costo, rendimiento, trazabilidad y capacidad de operar en contextos con restricciones de datos, aprovechando *RAG* para anclar las recomendaciones a evidencias verificables y minimizar alucinaciones.

Las garantías de calidad y seguridad se apoyarán en validación técnica automática (*JSON Schema*) y *QA* humano parcial para los *snippets*, trazabilidad explícita de cada recomendación hacia su fuente, y medidas de protección como minimización de datos, anonimización cuando proceda, cifrado en tránsito y en reposo, y controles de acceso basados en roles.

Para los próximos pasos, es necesario definir políticas claras sobre la generación y retención de reportes (*on-demand* vs almacenamiento persistente) dado que cada opción implica un intercambio de costos, trazabilidad y exposición de datos. Con esto, el prototipo a realizar debe seguir el siguiente flujo:

1. Ingestión de documentos.
2. Generación de snippets.
3. Lectura de recomendaciones usando RAG.
4. Creación de 10 a 50 reportes individuales personalizados que serán evaluados mediante *QA* humano.
5. Si el resultado está a la altura de lo esperado, experimentar con el uso de SLM locales para reducir los costos totales del proceso.

8. Anexo

- IBM. (s. f.). What is cybersecurity? IBM. https://www.ibm.com/think/topics/cybersecurity
- Kalhoro, S., Rehman, M., Ponnusamy, V. A., & Shaikh, F. B. (2021). Extracting Key Factors of Cyber Hygiene Behaviour Among Software Engineers: A Systematic Literature Review. IEEE Access, 9, 100781-100793.
- Nifakos, S., Chandramouli, K., Nikolaou, C. K., Papachristou, P., Koch, S., Panaousis, E., & Bonacina, S. (2021). Influence of Human Factors on Cyber Security within Healthcare Organisations: A Systematic Review. Sensors, 21(15), 5119.
- Maennel, K., Mäses, S., & Maennel, O. (2018). Cyber Hygiene: The Big Picture. En Secure IT Systems: 23rd Nordic Conference, NordSec 2018, Oslo, Norway, November 28–30, 2018, Proceedings (pp. 291–305). Springer.
- Vishwanath, A., Neo, L. S., Goh, P., Lee, S., Khader, M., Ong, G., & Chin, J. (2020). Cyber hygiene: The concept, its measure, and its initial tests. Decision Support Systems, 128, 113160.
- Fikry, A., Hamzah, M. I., Husseini, Z., Abdul, A. J., & Bakar, K. A. A. (2024). Defining the Beauty of Cyber Hygiene: A Retrospective Look. Proceedings of the 2024 International Conference on Cybersecurity and Digital Forensics.
- Kioskli, K., Fotis, T., Nifakos, S., & Mouratidis, H. (2023). The Importance of Conceptualising the Human-Centric Approach in Maintaining and Promoting Cybersecurity-Hygiene in Healthcare 4.0. Applied Sciences, 13(6), 3410.
- Belcak, P., Heinrich, G., Diao, S., Fu, Y., Dong, X., Muralidharan, S., Lin, Y. C., & Molchanov, P. (2025). Small Language Models are the Future of Agentic AI. arXiv. https://arxiv.org/abs/2506.02153
- Álvarez, C. (2025). Reporte HLT 2024-25 Ciberhigiene: Diagnóstico del estado de la ciberhigiene y la alfabetización digital en ciberseguridad del personal del Hospital Luis Tisné Brousse. Hospital Luis Tisné Brousse o Universidad de los Andes.
- Avendaño Palacios, C. M. (2025). HFShield: Plataforma para la gestión integral de factores humanos en ciberseguridad (Proyecto de título, Universidad de Los Andes, Facultad de Ingeniería y Ciencias Aplicadas, Santiago de Chile).
- Instituto Nacional de Ciberseguridad (INCIBE). (2024). *Instituto Nacional de Ciberseguridad (INCIBE)*. Recuperado de https://www.incibe.es/