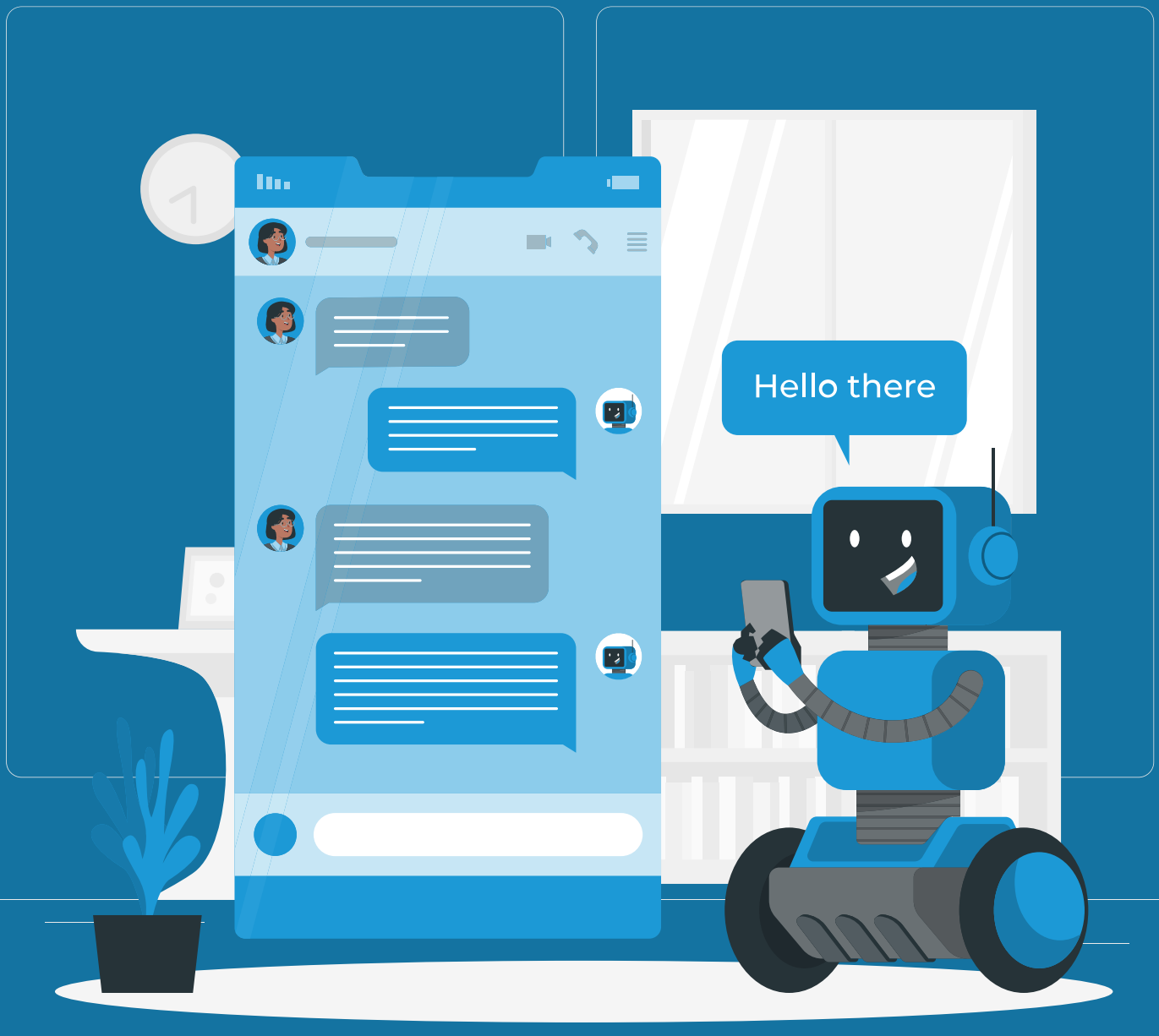
GOBIERNO DE ESPAÑA
MINISTERIO DE DEFENSA

CCN
centro criptológico nacional

CCN-TEC 014

Instalación y Securización de un **ChatBot** con LLM de forma local.

MARZO 2025



Edita:

GOBIERNO DE ESPAÑA
MINISTERIO DE DEFENSA

centro criptológico nacional

CENTRO CRIPTOLOGICO NACIONAL
cn=CENTRO CRIPTOLOGICO NACIONAL, 2.5.4.97=VATES-S2800155J, ou=CENTRO CRIPTOLOGICO NACIONAL, o=CENTRO CRIPTOLOGICO NACIONAL, c=ES
2025.03.28 09:19:04 +01'00'

Fecha de Edición: marzo de 2025

01.

SOBRE CCN

El Centro Criptológico Nacional (**CCN**) es el Organismo responsable de garantizar la seguridad de las Tecnologías de la Información y la Comunicación (**TIC**) en las diferentes entidades del Sector Público, así como la seguridad de los sistemas que procesan, almacenan o transmiten información clasificada.

Su ámbito de competencia está definido por el siguiente marco normativo:

- Ley 11/2002, 6 de mayo, reguladora del Centro Nacional de Inteligencia (CNI)
- Real Decreto 421/2004, 12 de marzo, que regula y define el ámbito y funciones del Centro Criptológico Nacional (CCN).
- Orden Ministerio Presidencia PRE/2740/2007, de 19 de septiembre, que regula el Esquema Nacional de Evaluación y Certificación de la Seguridad de las Tecnologías de la Información, y que confiere al CCN la capacidad de actuar como Organismo de Certificación (OC) de dicho Esquema.
- Real Decreto 03/2010, de 8 de enero, de desarrollo del Esquema Nacional de Seguridad (ENS), (actualizado en el RD 951/2015, de 23 de octubre) en el que se establecen los principios básicos y requisitos mínimos, así como las medidas de protección a implantar en los sistemas de la Administración.
- **Comisión Delegada del Gobierno para Asuntos de Inteligencia**, que define anualmente los objetivos del CNI mediante la Directiva de Inteligencia, que marca la actividad del Centro.

Otra normativa

Además de la legislación anteriormente expuesta, varias han sido las normativas publicadas en los dos últimos años que afectan muy directamente al sector de la ciberseguridad y, por ende, a la actividad del Centro Criptológico Nacional. Entre otras, podemos citar:

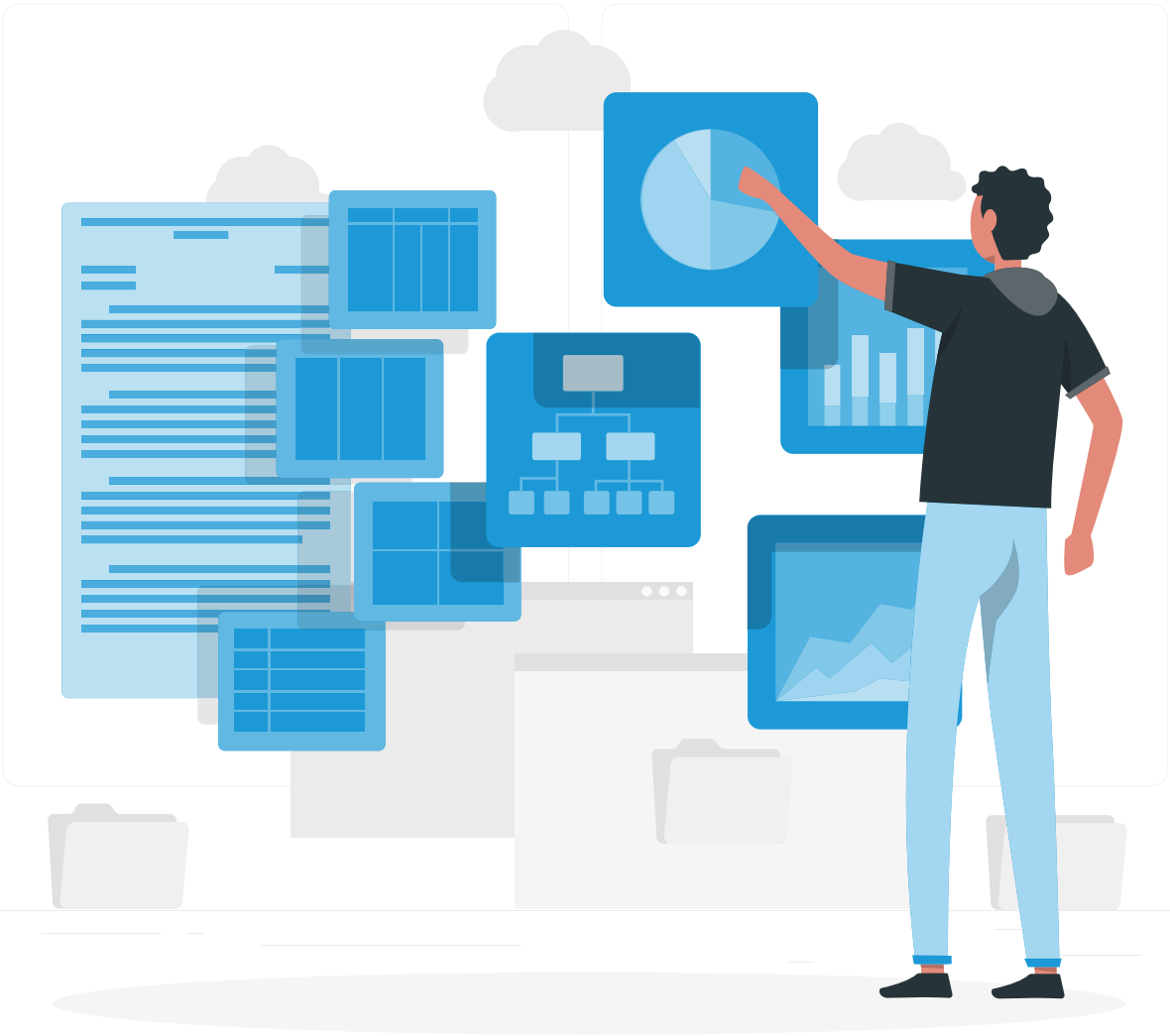
- Real Decreto-ley 14/2019, de 31 de octubre, por el que se adoptan medidas urgentes por razones de seguridad pública en materia de administración digital, contratación del sector público y telecomunicaciones
- La Estrategia Nacional de Ciberseguridad, aprobada el 30 de abril de 2019, desarrolla las previsiones de la Estrategia de Seguridad Nacional de 2017 en el ámbito de la ciberseguridad.
- Real Decreto-ley 12/2018, de 7 de septiembre, de seguridad de las redes y sistemas de información.
- Directiva (UE) 2016/1148 del Parlamento Europeo y del Consejo de 6 de julio de 2016 relativa a las medidas destinadas a garantizar un elevado nivel común de seguridad de las redes y sistemas de información en la Unión, también conocida como Directiva NIS. Se trata de la primera directiva comunitaria en materia de ciberseguridad, que entró en vigor el 9 de agosto de 2016 y que debe ser adoptada y publicada en cada uno de los Estados miembros antes del 9 de mayo de 2018.
- Estrategia de Ciberseguridad Nacional y los nueve Planes Derivados del Plan Nacional de Ciberseguridad, que fueron aprobados el 14 de julio de 2015 por el Consejo de Seguridad Nacional y que desarrollan la Estrategia y ordenan la actividad y el esfuerzo del Estado frente a este ámbito.
- Ley 39/2015, de 1 de octubre, del Procedimiento Administrativo Común de las Administraciones Públicas.
- Ley 40/2015, de 1 de octubre, de Régimen Jurídico del Sector Público, que amplía el ámbito de aplicación del ENS al sector público institucional y con ello la responsabilidad del CCN.
- Instrucción Técnica de Seguridad (ITS) de conformidad con el Esquema Nacional de Seguridad. Resolución de 13 de octubre de 2016 de la Secretaría de Estado de Administraciones Públicas.
- Instrucción Técnica de Seguridad (ITS) de Informe del Estado de la Seguridad. Resolución de 7 de octubre de 2016, de la Secretaría de Estado de Administraciones Públicas, por la que se aprueba la Instrucción Técnica de Seguridad de Informe del Estado de la Seguridad.
- Ley Orgánica 1/2015, de 30 de marzo, por la que se modifica la Ley Orgánica 10/1995, de 23 de noviembre, del Código Penal en materia de delitos de terrorismo incluyendo tipos penales de ciberterrorismo.
- Instrucción Técnica de Seguridad (ITS) de Notificación de Incidentes de Seguridad. Resolución de 13 de abril de 2018, de la Secretaría de Estado de Función Pública, por la que se aprueba la Instrucción Técnica de Seguridad de Notificación de Incidentes de Seguridad.
- Instrucción Técnica de Seguridad (ITS) de Auditoría de la Seguridad de los Sistemas de Información. Resolución de 27 de marzo de 2018, de la Secretaría de Estado de Función Pública, por la que se aprueba la Instrucción Técnica de Seguridad de Auditoría de la Seguridad de los Sistemas de Información.
- Instrucción Técnica de Seguridad (ITS) de conformidad con el Esquema Nacional de Seguridad Resolución de 13 de octubre de 2016, de la Secretaría de Estado de Administraciones Públicas, por la que se aprueba la Instrucción Técnica de Seguridad de conformidad con el Esquema Nacional de Seguridad.

02.

INTRODUCCIÓN

En los últimos años, los Modelos de Lenguaje de Gran Tamaño (**LLM**, por sus siglas en inglés de *Large Language Model*) han cambiado radicalmente la manera en que interactuamos con la inteligencia artificial (**IA**). Gracias a su entrenamiento con enormes volúmenes de datos y el uso de redes neuronales profundas, estos modelos pueden generar texto de forma fluida, responder preguntas y hasta imitar distintos estilos de escritura. Sin embargo, su implementación no está exenta de riesgos.

Desde la propagación de información falsa hasta la presencia de sesgos algorítmicos, además de la preocupación sobre la privacidad y seguridad, el uso inadecuado de los **LLM** puede traer consecuencias no deseadas.



03.

OBJETO

Los objetivos del presente documento son:

- Detallar la implementación de un **LLM** local con una interfaz gráfica para la interacción y configuración de modelos de lenguaje.
- Comentar los principales riesgos de seguridad y explicar las contramedidas necesarias para su mitigación.

04.

IMPLEMENTACIÓN DEL CHATBOT MEDIANTE LLM

04.1. Plataforma de gestión de modelos de inteligencia artificial

Para la instalación del *ChatBot* en primer lugar, será necesario instalar una herramienta que permita la gestión, descarga y ejecución de los distintos modelos de inteligencia artificial que se consideren. En este caso se propone la herramienta *Ollama*, disponible tanto en *Windows*, como *macOS* y *Linux*. **(FIGURA 1)**

Para su descarga se debe acudir a la web oficial de la herramienta[1] y descargar el binario de la plataforma correspondiente que se vaya a utilizar y ejecutarlo.

En este caso, se procede a instalar la versión de *Windows* accediendo a la sección de descargas de la mencionada página web y descargando el ejecutable correspondiente. **(FIGURA 2)**

Tras la instalación del gestor de *Ollama*, se podrá iniciar pulsando sobre el icono de la aplicación. **(FIGURA 3)**

Una vez iniciado, aparecerá un icono en la esquina inferior derecha. **(FIGURA 4)**

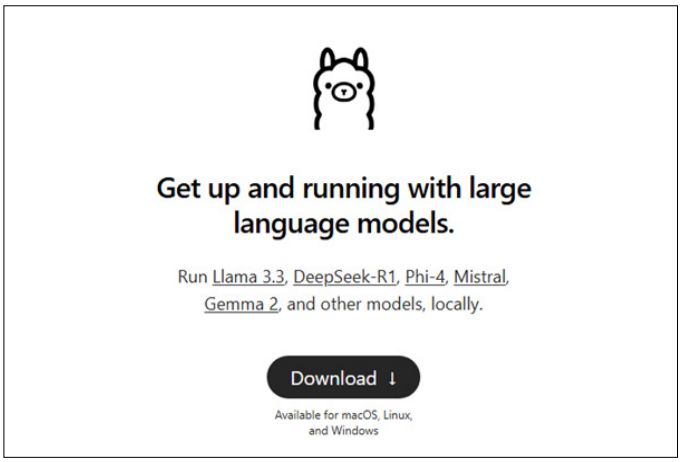


FIGURA 1. WEB DE DESCARGA DE OLLAMA

FIGURA 2. DESCARGA DE OLLAMA

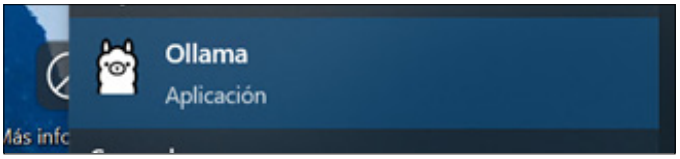


FIGURA 3. APLICACIÓN OLLAMA

FIGURA 4. ICONO OLLAMA

[1] https://ollama.com/

El siguiente paso es seleccionar el modelo **LLM** que se desee utilizar. Para ello, se puede acceder a la sección de búsqueda de la web de *Ollama*[2] en donde aparecen todos los modelos disponibles. **(FIGURA 5)**

El modelo puede ser descargado mediante la ejecución del siguiente comando desde una consola de Windows: **ollama pull <modelo_elegido>**

Por ejemplo **(FIGURA 6)**.

Tras la descarga del modelo, se puede ejecutar mediante el siguiente comando: **ollama run <modelo_elegido>**

Es posible tener varios modelos descargados y ejecutar el que se considere en cada momento.

Tras la ejecución del comando, se podrá interactuar con el modelo desde la misma consola **(FIGURA 7)**.

Una vez completados estos pasos, ya se dispone de un *ChatBot* funcional utilizando el modelo de **LLM** elegido.

No obstante, con el objetivo de realizar la interacción de una manera más cómoda y tener la capacidad de añadir otras funcionalidades al sistema de manera sencilla, a continuación, se explica el proceso de instalar un sistema que permitirá gestionar de manera gráfica las funcionalidades y la interacción con el modelo.

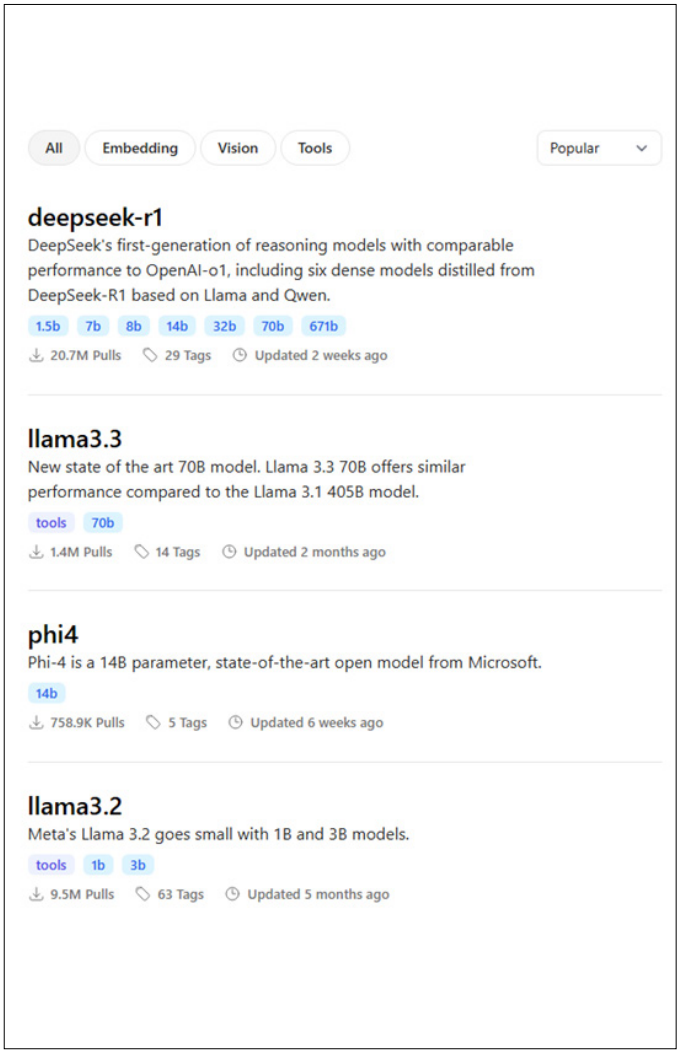


FIGURA 5. LISTA DE MODELOS DISPONIBLES

```
C:\Users\user>ollama pull llama3.3
pulling manifest
pulling 4824460d29f2... 100% ▕[redacted]▏  42 GB
pulling 948af2743fc7... 100% ▕[redacted]▏ 1.5 KB
pulling bc371a43ce90... 100% ▕[redacted]▏ 7.6 KB
pulling 53a87df39647... 100% ▕[redacted]▏ 5.6 KB
pulling 56bb8bd477a5... 100% ▕[redacted]▏  96 B
pulling c7091aa45e9b... 100% ▕[redacted]▏ 562 B
verifying sha256 digest
writing manifest
success
```

FIGURA 6. DESCARGA DEL MODELO

```
C:\Users\user>ollama run llama3.2
>>> Hola!
¡hola! ¿En qué puedo ayudarte hoy?

>>> Send a message (/? for help)
```

FIGURA 7. INTERACCIÓN POR CONSOLA LLM

[2] https://ollama.com/search

04.2. Instalación de la interfaz web de consulta

La solución que se propone en este caso es *OpenWeb UI*[3]. *OpenWeb UI* es una interfaz de usuario de código abierto diseñada para facilitar la interacción con modelos de lenguaje y gestionar la configuración de aplicaciones basadas en inteligencia artificial.

Existen diversas maneras de instalar *OpenWeb UI*, en este caso se indican los pasos de instalación utilizando *Docker Desktop* en *Windows* con el objetivo de simplificar el proceso.

Para ello desde la página oficial[4] se procede a descargar la herramienta *Docker* en primer lugar. **(FIGURA 8)**.

Tras la descarga del ejecutable de la versión de la plataforma correspondiente, se deberá ejecutar para comenzar la instalación de *Docker Desktop* **(FIGURA 9)**.

Una vez instalado *Docker*, tras ejecutarlo, se pulsará sobre la opción de terminal que aparece en la esquina inferior derecha **(FIGURA 10)**.

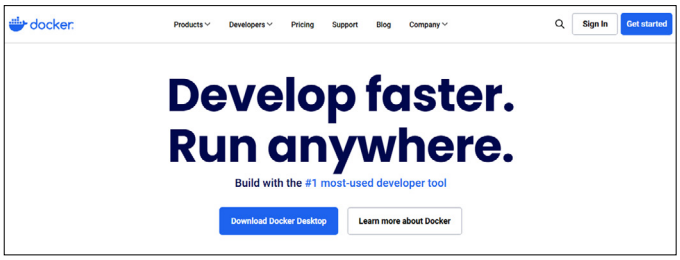


FIGURA 8. PÁGINA OFICIAL DOCKER

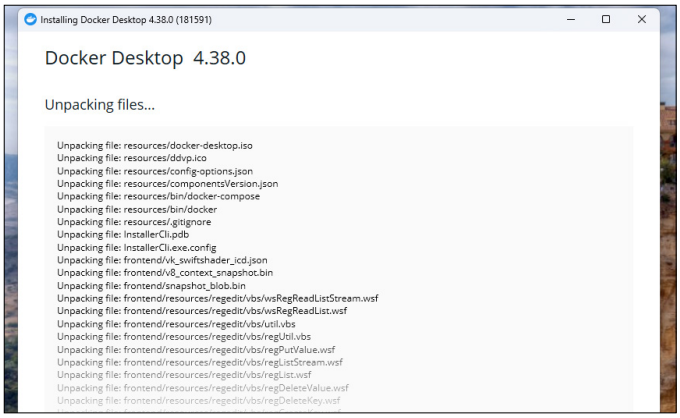


FIGURA 9. INSTALACIÓN DE DOCKER

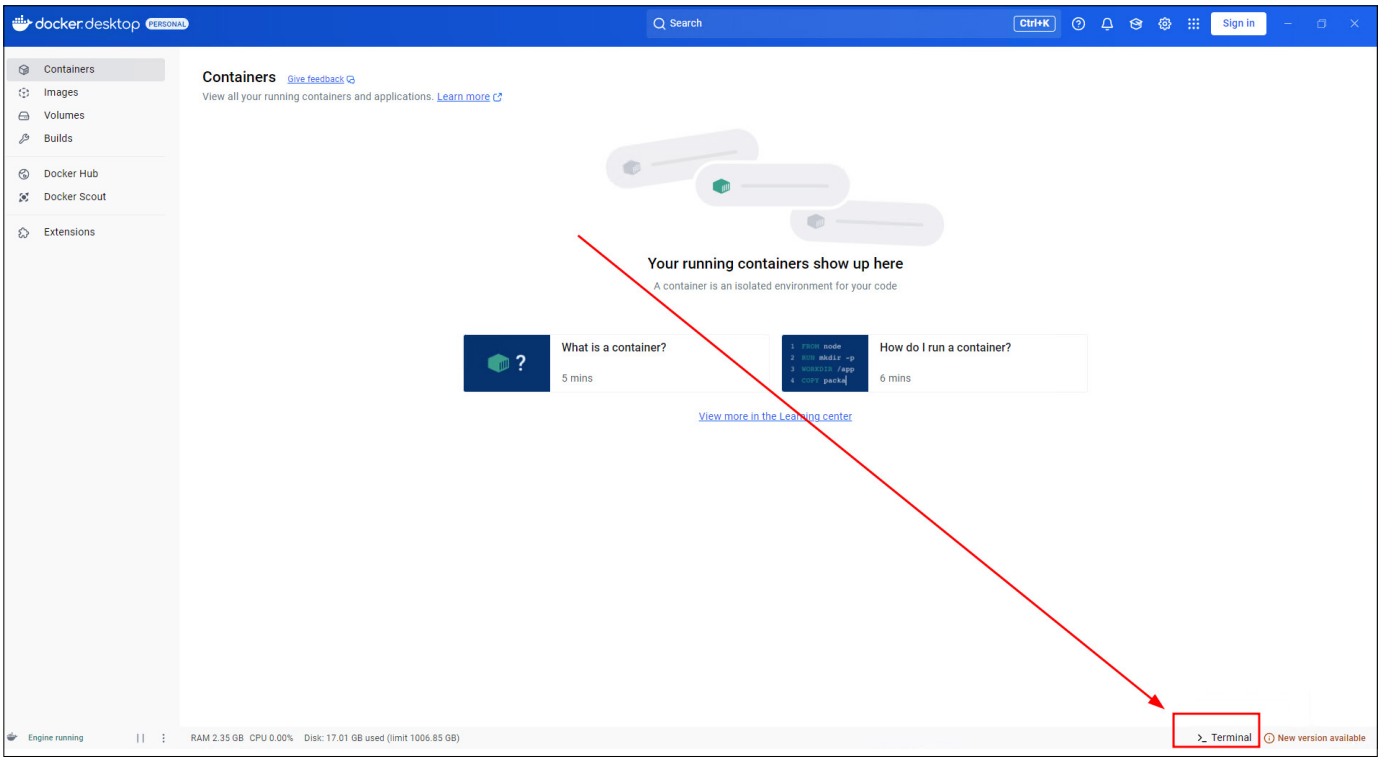


FIGURA 10. TERMINAL DOCKER WINDOWS

[3] https://openwebui.com/

[4] https://www.docker.com

En la terminal se deberán introducir los siguientes comandos:

01 | Para la descarga de la imagen oficial de *OpenWeb UI* desde su repositorio, se ejecutará el siguiente comando: **docker pull ghcr.io/open-webui/open-webui:main**

En el caso de que se disponga de un equipo con tarjeta gráfica dedicada se deberá utilizar una imagen optimizada para esa tecnología. Para su descargar se ejecutará el siguiente comando: **docker pull ghcr.io/open-webui/open-webui:cuda**

(FIGURA 11).

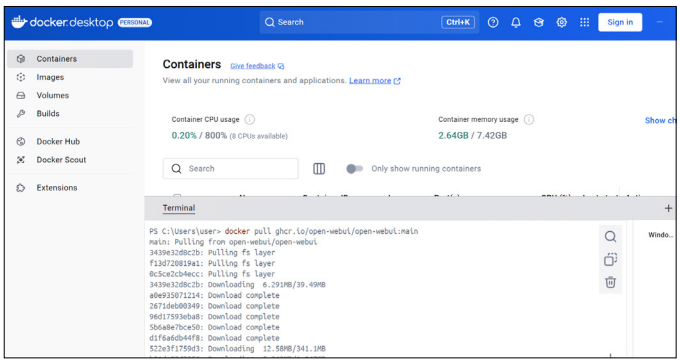


FIGURA 11. DESCARGA DE IMAGEN DOCKER OPENWEB UI

02 | Tras la descarga de la imagen se ejecutará el siguiente comando para iniciar la aplicación utilizando la imagen seleccionada: **docker run -d -p 3000:8080 -v open-webui:/app/backend/data --name open-webui ghcr.io/open-webui/open-webui:main**

Si se dispone de tarjetas gráficas dedicadas se puede utilizar el siguiente comando: **docker run -d -p 3000:8080 -v open-webui:/app/backend/data --gpus all --name open-webui ghcr.io/open-webui/open-webui:cuda**

(FIGURA 12).

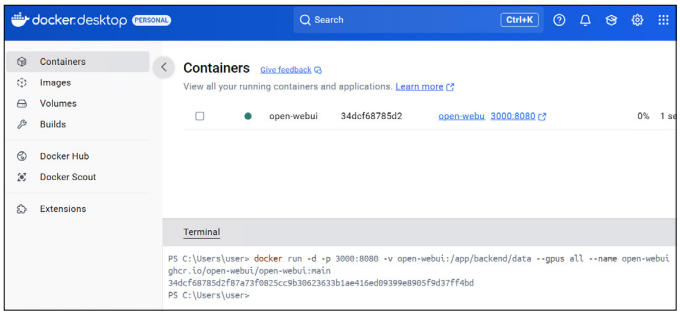


FIGURA 12. INSTALACIÓN OPENWEB UI

A continuación, se explican las acciones que realiza el comando anterior, pudiéndose personalizar en cada caso en base a las características de la instalación:

- **docker run:** este es el comando principal para ejecutar un contenedor *Docker*.
- **-d:** ejecuta el contenedor en segundo plano (modo *detached*).
- **-p 3000:8080:** mapea el puerto 3000 del *host* al puerto 8080 del contenedor. Esto permite acceder al servicio en el contenedor a través del puerto 3000 del *host*.
- **-v open-webui:/app/backend/data:** monta un volumen llamado *open-webui* en el directorio */app/backend/data* dentro del contenedor. Esto permite que los datos sean persistentes entre ejecuciones del contenedor.
- **--gpus all:** asigna todas las GPUs disponibles al contenedor. Esto es útil si el contenedor necesita realizar operaciones que requieren aceleración por GPU.
- **--name open-webui:** asigna el nombre *open-webui* al contenedor. Esto facilita la gestión del contenedor, ya que puede ser referido a él por su nombre.
- **ghcr.io/open-webui/open-webui:cuda:** especifica la imagen Docker que se va a utilizar para crear el contenedor. En este caso, la imagen se encuentra en el registro de *GitHub Container Registry* (GHCR).

Se pueden consultar más detalles sobre la instalación de *OpenWeb UI* mediante *Docker* en la documentación oficial de la herramienta[5].

[5] https://docs.openwebui.com/getting-started/quick-start/

Una vez completados los pasos indicados, en la sección ***Containers*** aparecerá el contenedor de ***OpenWeb UI*** con el nombre que se le haya asignado **(FIGURA 13)**.

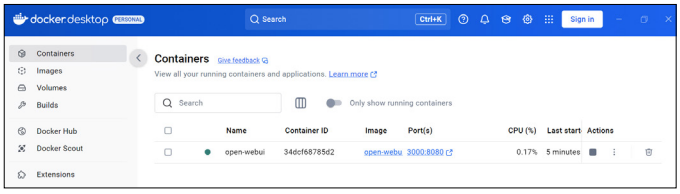


FIGURA 13. CONTENEDORES DOCKER EN EJECUCIÓN

03 | Para acceder al interfaz web del ChatBot bastará con acceder a la siguiente URL desde un navegador **(FIGURA 14)**: http://localhost:3000.

En este punto ya tendríamos un ChatBot funcional utilizando el modelo de LLM elegido y con un entorno gráfico amigable.

FIGURA 14. INTERFAZ WEB PARA INTERACCIÓN CON EL MODELO

04.3. Personalización de modelos

OpenWeb UI permite configurar multitud de opciones en el modelo añadiendo parámetros adicionales. Es posible establecer filtros o segmentar el acceso a determinados usuarios, entre otras acciones. Para ello se debe acceder a la sección de espacio de trabajo y seleccionar la pestaña modelos **(FIGURA 15)**.

Y seleccionar el modelo a configurar **(FIGURA 16)**.

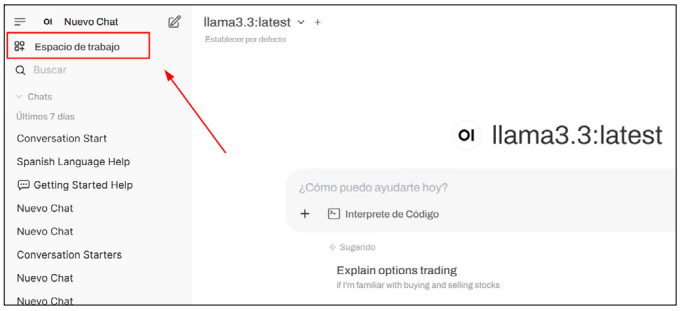


FIGURA 15. ACCESO A ESPACIO DE TRABAJO

FIGURA 16. ACCESO A CONFIGURACIÓN DEL MODELO

Tras seleccionar el modelo en concreto que se quiere personalizar, se despliega la pestaña Parámetros avanzados **(FIGURA 17)**.

Dentro de la sección Parámetros avanzados existen distintas opciones para configurar de manera precisa el comportamiento del modelo mediante un conjunto de parámetros que modifican la manera en la que responde.

Parámetros como la temperatura controlan el equilibrio entre precisión y creatividad, mientras que los ajustes de *Top P* / *Top K* determinan la coherencia general de las respuestas generadas.

Asimismo, es posible establecer penalizaciones de frecuencia y presencia para reducir repeticiones, así como limitar la longitud máxima de respuesta mediante el parámetro *max tokens* **(FIGURA 18)**.

OpenWeb UI también implementa un sistema de configuración vía URL que permite predefinir qué modelo utilizar, activar búsqueda web automática, habilitar herramientas específicas e incluso establecer consultas iniciales para sesiones sin necesidad de ajustes manuales.

Por otro lado, recorriendo la pestaña superior es posible añadir una base de datos de conocimiento a nuestro modelo o establecer un *prompt* de contexto predeterminado **(FIGURA 19)**.

Los usuarios pueden crear bases de conocimiento dedicadas mediante la carga de diversos formatos de documentos que son procesados e indexados de manera automática y personalizable. Estas bases de conocimiento pueden asignarse a modelos específicos, permitiéndoles acceder y referenciar información contextual al generar respuestas.

Complementando esta funcionalidad, *OpenWeb UI* ofrece integración con múltiples proveedores de búsqueda web como por ejemplo *SearXNG*, *Google PSE*, *Brave*, *Serpstack* y *Serper*, lo que permite a los modelos incorporar información actualizada de Internet en tiempo real.

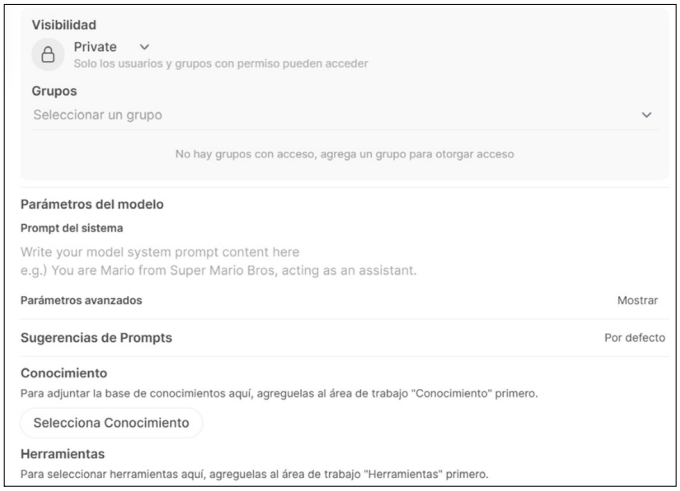


FIGURA 17. CONFIGURACIÓN DE LOS PARÁMETROS DEL MODELO

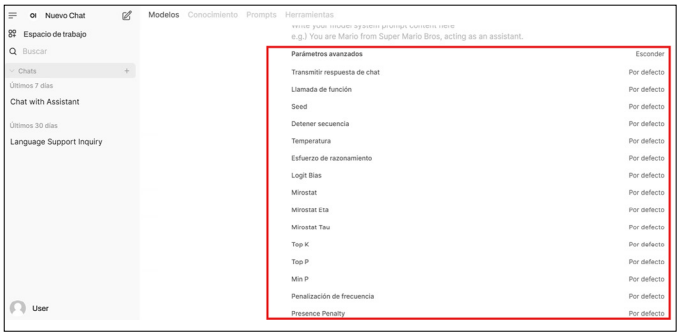


FIGURA 18. PARÁMETROS AVANZADOS PARA MODELOS

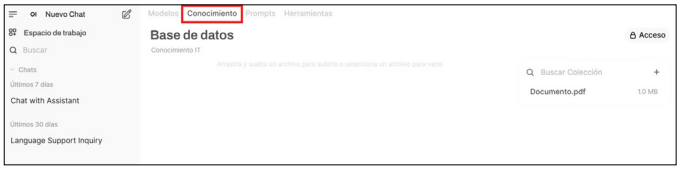
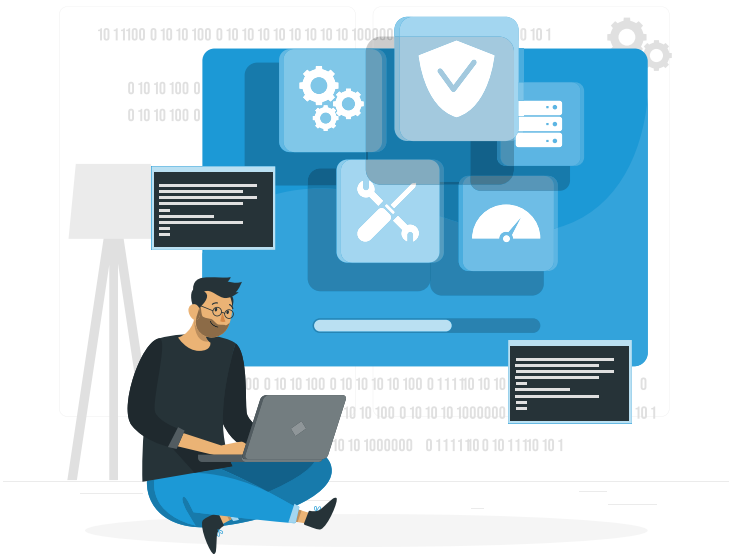


FIGURA 19. BASES DE DATOS DE CONOCIMIENTO



05.

SEGURIDAD EN MODELOS LLM

Desde la generación de información errónea hasta la reproducción de sesgos algorítmicos, pasando por riesgos relacionados con la privacidad y la seguridad, los Modelos de Lenguaje de Gran Tamaño (**LLM**) presentan diversas vulnerabilidades si no se regulan y supervisan adecuadamente.

Uno de los principales desafíos es su capacidad para generar contenido falso o engañoso. Asimismo, dado que estos modelos se entrenan con grandes volúmenes de datos, pueden heredar y amplificar sesgos preexistentes, lo que podría derivar en respuestas parciales.

En términos de seguridad, los **LLM** son susceptibles a ataques, donde actores malintencionados pueden explotar sus respuestas para manipular información, generar contenido dañino o acceder a datos sensibles. Además, el uso de enormes cantidades de información para su entrenamiento plantea retos en cuanto a la privacidad, especialmente cuando los datos incluyen información personal o confidencial sin la debida protección.

Por ello, resulta esencial implementar medidas que mitiguen estos riesgos, promoviendo mayor transparencia en su funcionamiento, garantizando la precisión en sus respuestas y reforzando la seguridad en su desarrollo y aplicación.

En el ámbito de la ciberseguridad, la *Open Web Application Security Project* (**OWASP**) es una organización reconocida por identificar y clasificar vulnerabilidades en sistemas y aplicaciones.

Con el crecimiento del uso de Modelos de Lenguaje de Gran Tamaño (**LLM**), **OWASP** ha desarrollado una guía[6] con los 10 principales riesgos asociados a estos modelos, con el objetivo de concienciar sobre sus vulnerabilidades y fomentar mejores prácticas en su desarrollo e implementación.

A continuación, se explican los principales puntos que detalla **OWASP** y qué componentes se pueden configurar para reducir los distintos riesgos que se generan.

[6] https://owasp.org/www-project-top-10-for-large-language-model-applications/

05.1. Inyección de *prompts*

La inyección de *prompts* es un tipo de ciberataque dirigido específicamente contra los Modelos de Lenguaje de Gran Tamaño (**LLM**). En estos ataques, los atacantes enmascaran entradas maliciosas como *prompts* legítimos, manipulando así los sistemas de inteligencia artificial (**IA**) generativa para que filtren datos confidenciales, difundan información errónea o realicen acciones no autorizadas.

Esta vulnerabilidad explota una característica fundamental de los **LLM**: su incapacidad para distinguir claramente entre las instrucciones del desarrollador (*prompts* del sistema) y las entradas del usuario.

El mecanismo de funcionamiento de este ataque aprovecha que tanto los *prompts* del sistema como las entradas de los usuarios tienen el mismo formato: cadenas de texto en lenguaje natural.

Existen principalmente dos variantes de este ataque:

- **Inyección directa:** los atacantes controlan directamente la entrada del usuario y alimentan el *prompt* malicioso enviado al **LLM**.
- **Inyección indirecta:** los atacantes ocultan sus acciones en datos que el **LLM** podría consumir, como colocar mensajes en páginas web que el **LLM** leería. También pueden incrustarse en imágenes o ficheros que el **LLM** escanea.

OpenWeb UI ofrece varias características específicas que ayudan a reducir el riesgo de inyección de *prompts*.

Es posible, por ejemplo, establecer *prompts* que definan claramente las limitaciones del modelo. A través de su interfaz de edición de modelos se pueden establecer restricciones sobre qué tipo de instrucciones debe ignorar el modelo.

Además, *OpenWeb UI* incorpora un sistema de *pipelines*, descrito como *UI-Agnostic OpenAI API Plugin Framework*, que permite, entre otras acciones, interceptar y procesar los *prompts* antes de que lleguen al **LLM**.

Este sistema incluye:

- **Filtros:** procesan todos los prompts independientemente del modelo seleccionado, lo que permite implementar mecanismos de validación de manera global.
- **Pipes:** son módulos seleccionables por el usuario para procesamiento específico de *prompts*.

Este sistema puede configurarse para detectar y bloquear patrones de inyección conocidos antes de que lleguen al modelo, funcionando como una capa adicional de seguridad.

Para instalar la funcionalidad de *pipelines* se deberá ejecutar en la terminal de *Docker Desktop* el siguiente comando: **docker run -d -p 9099:9099 --add-host=host.docker.internal:host-gateway -v pipelines:/app/pipelines --name pipelines --restart always ghcr.io/open-webui/pipelines:main**

(FIGURA 21).

Cuando se ejecute el comando se creará un nuevo contenedor *pipelines* desde donde se podrán gestionar las opciones comentadas anteriormente.

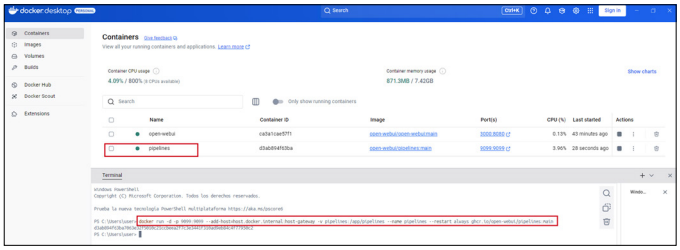
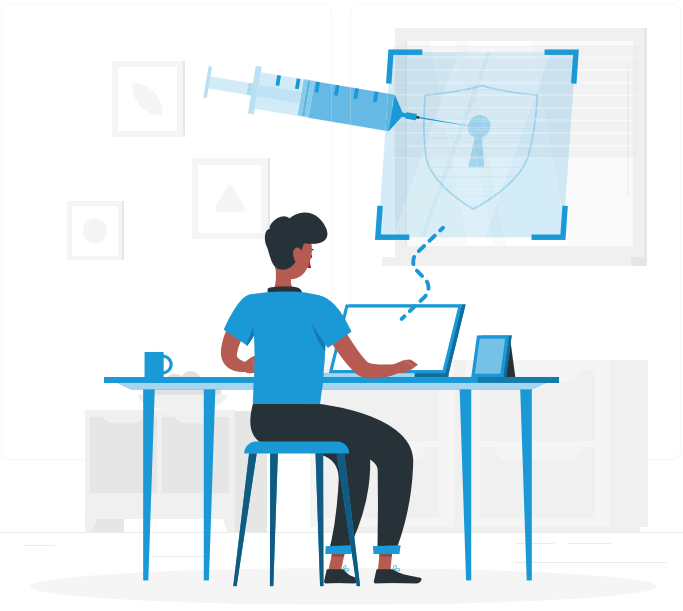


FIGURA 21. INSTALACIÓN DE PIPELINES



Para configurar este nuevo servicio se deberá acudir a la interfaz de *OpenWeb UI*, accediendo al menú de configuración tal y como se indica en la siguiente imagen **(FIGURA 22)**.

Posteriormente se deberá acceder a la opción Gestionar conexiones *API* de *OpenAI* y tras pulsar sobre el símbolo + añadir los datos que se muestran en la siguiente imagen para configurar la conexión entre el contenedor base de *OpenWeb UI* y el de *pipelines* **(FIGURA 23)**.

Una vez añadida la conexión se accede a la sección *Pipelines* para poder añadir elementos a nuestro sistema **(FIGURA 24)**.

Desde dicha sección ya se podrán instalar los elementos que se consideren. A modo de ejemplo en el *Git*[7] de *Open WebUI* se pueden encontrar algunos *pipelines* y filtros para utilizar. *Pipelines* está diseñado para balancear cargas de trabajo complejas de la instancia principal. Si se quiere instalar filtros básicos existe una opción más accesible, denominada funciones[8] de *OpenWeb UI*.

La siguiente imagen muestra la configuración para instalar un filtro para reducir el riesgo del ataque *prompt injection* **(FIGURA 25)**.

Dentro de la funcionalidad es posible utilizar un *script* en *Python* o referenciar la URL de un repositorio. Se puede obtener más detalle de estas funcionalidades en la documentación oficial de la herramienta[9].

También es posible ajustar los parámetros del modelo para prevenir este tipo de ataques. Se recomienda establecer una temperatura más baja, lo que hace que el modelo sea más determinista y menos propenso a seguir instrucciones maliciosas o aumentar la penalización por frecuencia, lo que puede evitar que el modelo repita instrucciones de sistema que un atacante intenta anular.

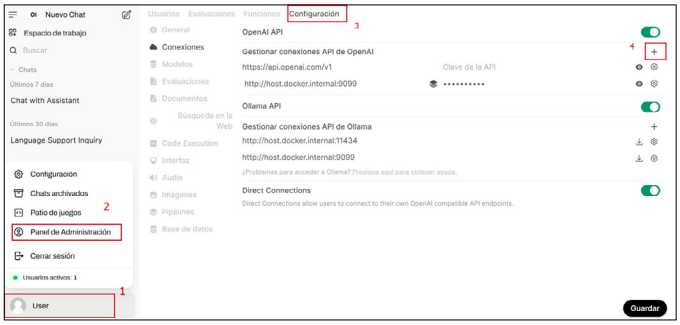


FIGURA 22. CONFIGURACIÓN DE CONEXIONES

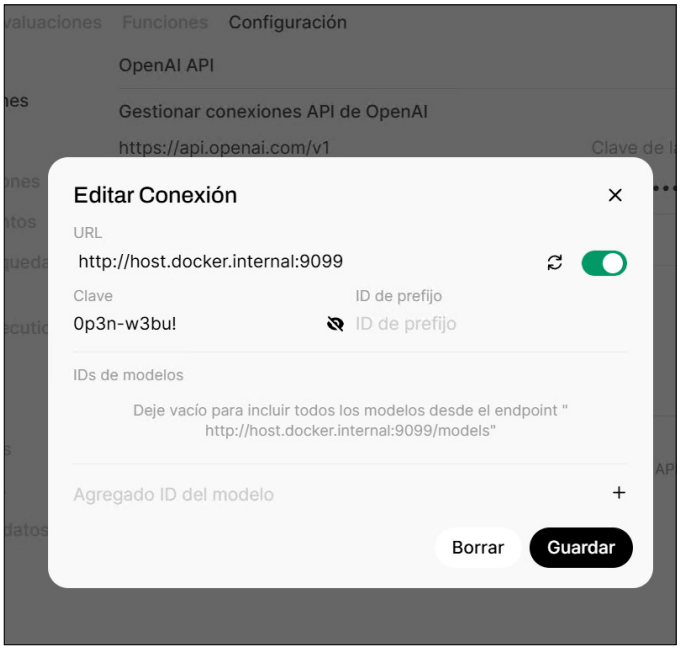


FIGURA 23. CONFIGURACIÓN DE PIPELINES

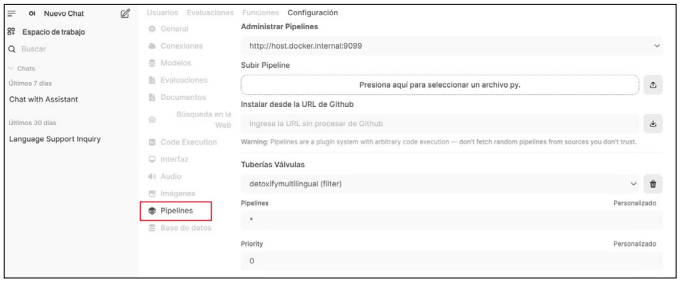


FIGURA 24. ADMINISTRACIÓN DE PIPELINES

FIGURA 25. CONFIGURACIÓN PARA INSTALAR UN FILTRO

[7] https://github.com/open-webui/pipelines

[8] https://docs.openwebui.com/features/plugin/functions

[9] https://docs.openwebui.com/pipelines/

05.2. Divulgación de información sensible

Este riesgo consiste en la revelación de datos sensibles en las respuestas de los modelos, exponiendo potencialmente información protegida como datos personales, credenciales de seguridad o información del sistema.

En el caso de las instalaciones *offline* todos los datos, *prompts* y resultados permanecen dentro de un entorno controlado, reduciendo riesgos de transmisión no autorizada hacia servidores externos y previniendo escenarios de filtración cruzada. A diferencia de soluciones SaaS de **IA** donde múltiples clientes comparten infraestructura, este aislamiento reduce las vulnerabilidades asociadas con ese tipo de arquitecturas.

Además, es posible implementar modelos optimizados para seguridad, gestionar el ciclo completo de los datos utilizados y establecer controles granulares sobre quién puede acceder a modelos específicos según roles.

Para evitar tanto la inyección de *prompts* como la divulgación de información sensible existen sistemas como por ejemplo *Llama Guard* o *LLM-Guard*.

Llama Guard es un modelo especializado de seguridad desarrollado por Meta, diseñado específicamente para reforzar la seguridad de los **LLM**. A diferencia de los modelos generativos tradicionales, *Llama Guard*[10,11] funciona como sistema de clasificación de seguridad que evalúa si las entradas o salidas de un **LLM** contienen contenido potencialmente malicioso.

Este modelo está entrenado para detectar y clasificar más de 20 categorías de riesgo específicas, incluyendo contenido ilegal, acoso, incitación al odio, información confidencial y otros tipos de contenido inapropiado.

En *OpenWeb UI*, *Llama Guard* puede implementarse como componente integral del sistema de *Pipelines*, funcionando como filtro de seguridad o como función específica dentro de otro modelo.

La implementación más común se basa en configurar *Llama Guard* como modelo adicional que opera en paralelo con el **LLM** principal, escaneando automáticamente tanto *prompts* de usuario como respuestas generadas para identificar contenido malicioso.

Cuando se detecta contenido potencialmente inadecuado, *OpenWeb UI* puede configurarse para bloquear la respuesta, generar advertencias, o solicitar confirmación adicional del usuario. Esta integración que permite incorporar múltiples modelos y herramientas en un entorno controlado.

A modo de ejemplo se indica los pasos de instalación de *Llama Guard* como filtro mediante funciones de *OpenWeb UI*:

01 | Descargar el modelo de filtrado y moderación mediante la ejecución del siguiente comando en una consola de Windows: **ollama pull llama-guard3:latest**

02 | Acceder a la sección funciones dentro del panel de administración para configurar la función deseada **(FIGURAS 26.1 y 26.2)**.

FIGURA 26.1. SELECCIÓN MENÚ DE USUARIO

FIGURA 26.2. ACCESO A FUNCIONES

[10] https://ai.meta.com/research/publications/llama-guard-llm-based-input-output-safeguard-for-human-ai-conversations/

[11] https://huggingface.co/meta-llama/Llama-Guard-3-8B

Desde dicha sección se podrán añadir funciones que hagan uso del modelo de moderación *Llama Guard* pulsando sobre el símbolo +.

Las funciones se pueden obtener desde la página oficial de *OpenWeb UI* en el apartado funciones[12]. La siguiente imagen muestra un ejemplo de una función obtenida desde la web mencionada **(FIGURA 27)**.

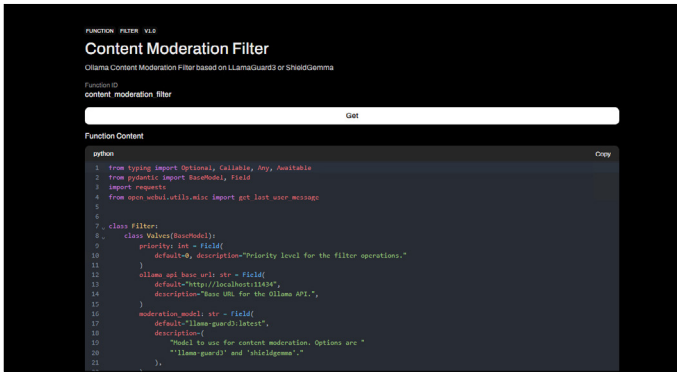


FIGURA 27. FUNCIÓN OBTENIDA WEB OPENWEBUI

03 | Copiar y pegar el código de la función en el menú de funciones de *OpenWeb UI* que se ha accedido anteriormente **(ver FIGURA 26.2)** para que aparezca como función en la plataforma **(FIGURA 28)**.

FIGURA 28. FUNCIÓN EN OPENWEB UI

04 | Se configura la función pulsando en el icono del engranaje **(FIGURA 29)**.

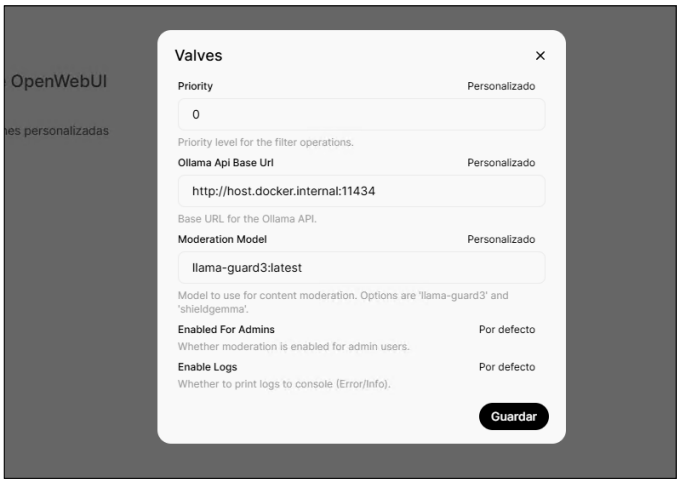


FIGURA 29. CONFIGURACIÓN FUNCIÓN

05 | Por último, se activa la función añadida **(FIGURA 30)**.

FIGURA 30. ACTIVACIÓN DE LA FUNCIÓN

[12] https://www.openwebui.com/functions

05.3. Vulnerabilidades en la cadena de suministro

Las vulnerabilidades en cadena de suministro introducen un riesgo cuando un **LLM** no está adecuadamente aislado de sus componentes, exponiendo la aplicación a compromisos a través de bibliotecas, modelos pre-entrenados o datos utilizados durante el entrenamiento.

Este tipo de vulnerabilidad no solo afecta a las dependencias de los componentes que se utilicen si no que puede llegar a afectar a los datos de entrenamiento de los modelos base utilizados.

Al implementar *OpenWeb UI* en contenedores *Docker* aislados con dependencias explícitamente definidas, se establece un entorno de ejecución que minimiza la superficie de ataque, previniendo conexiones no autorizadas que podrían realizar componentes comprometidos.

Por otro lado, el sistema de *Pipelines* de *OpenWeb UI* ofrece mecanismos para implementar verificaciones de seguridad continuas, permitiendo la creación de filtros personalizados para validar la integridad y procedencia de cada componente.

Esta capacidad se puede complementar con procedimientos para la gestión de modelos y dependencias, creando un inventario estructurado con todos los componentes utilizados mediante un enfoque similar al **SBOM** (*Software Bill of Materials*)[13].

Para reforzar estas protecciones, se recomienda implementar un proceso formal de aprobación para incorporar nuevos modelos que incluya verificación criptográfica, evaluación de seguridad limitando el uso exclusivamente a modelos y repositorios verificados.

[13] https://cisa.gov/sbom

05.4. Envenenamiento de Datos y Modelos

Este riesgo se introduce cuando los datos de pre-entrenamiento, *fine-tuning* o *embeddings* son alterados maliciosamente para generar vulnerabilidades como puertas traseras o sesgos en el modelo.

En un ataque de envenenamiento los atacantes manipulan deliberadamente los datos de entrenamiento de un modelo de **IA** para comprometer su seguridad, eficacia o comportamiento.

Este tipo de ataques pueden manifestarse de diversas formas, siendo las más comunes:

- **FrontRunning Data Poisoning:** los atacantes identifican el momento en que se realizan capturas de datos para el entrenamiento e introducen manipulaciones maliciosas justo antes.
- **Split-View Data Poisoning:** implica la manipulación de contenido indexado que será utilizado para entrenamiento, comprometiendo la integridad de los *datasets*.
- **Inyección de Datos:** los atacantes introducen llamadas de datos maliciosos en los conjuntos de entrenamiento para hacer que el modelo produzca resultados específicos cuando encuentre ciertos desencadenantes.
- **Ataques de Puerta Trasera:** se introducen patrones específicos en los datos de entrenamiento que posteriormente pueden activarse para manipular el comportamiento del modelo.

Para reducir estos riesgos, se debe verificar criptográficamente la procedencia e integridad de todos los datos mediante técnicas de hash *fingerprinting*.

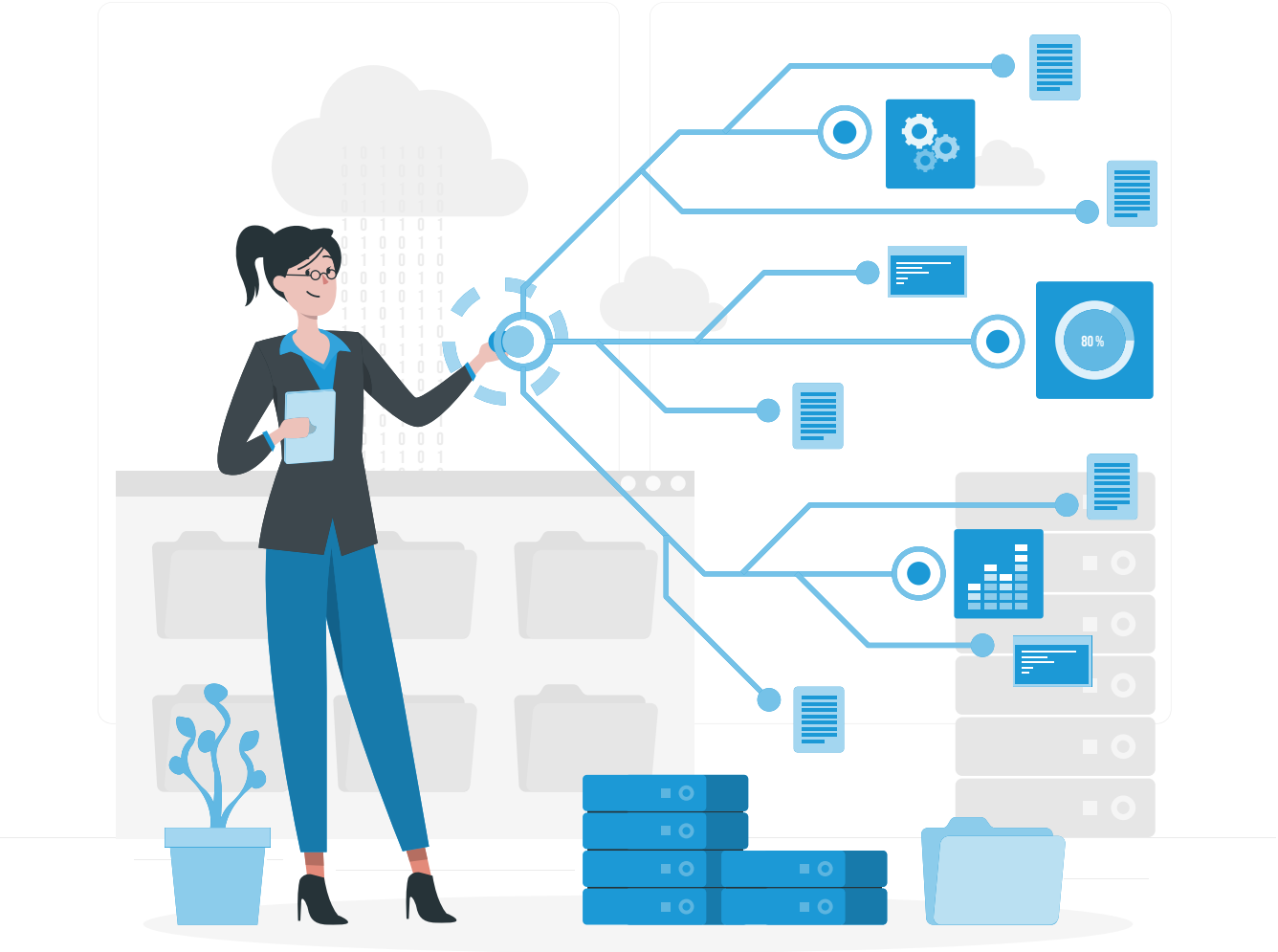
Además, el sistema **RAG** (*Retrieval Augmented Generation*) integrado en *OpenWeb UI* ofrece una capa de seguridad adicional al permitir al desarrollar gestionar cómo los modelos acceden y utilizan información.

Esta tecnología (**RAG**) combina la capacidad generativa de los **LLM** con un sistema de recuperación de información que consulta bases de conocimiento en tiempo real, permitiendo que el modelo responda basándose en datos introducidos por el administrador en lugar de depender exclusivamente del conocimiento memorizado durante su entrenamiento.

Esta capacidad permite vincular modelos **LLM** con repositorios de datos controlados por el administrador del sistema proporcionando acceso a información sin exposición a fuentes potencialmente comprometidas. Al reducir la dependencia del conocimiento memorizado durante el entrenamiento, este sistema **RAG** minimiza el impacto potencial de ataques de envenenamiento previos.

Se puede consultar más información sobre esta capacidad en la documentación oficial de la herramienta[14].

[14] https://docs.openwebui.com/tutorials/tips/rag-tutorial



05.5. Gestión inadecuada de resultados

La gestión inadecuada de resultados o salidas ocurre cuando se muestran resultados del modelo sin la debida validación, lo que puede permitir la ejecución de código malicioso o filtración de datos sensibles.

Esta vulnerabilidad se materializa cuando una aplicación utiliza directamente la salida de un **LLM** en operaciones sensibles como comandos de sistema, consultas de base de datos o generación de código sin una verificación adecuada.

Como mitigación principal se recomienda la implementación del sistema de *Pipelines* o filtros, comentado anteriormente que permitan interceptar y validar las respuestas del modelo antes de su presentación al usuario o su ejecución.

Asimismo, la inclusión de modelos de seguridad como *Llama Guard*[15] en estos filtros permite disminuir el riesgo ante este tipo de ataques. Ya que, al implementar esta herramienta como modelo de clasificación de seguridad, se obtiene una capa adicional especializada en detectar y filtrar contenido potencialmente dañino.

Llama Guard y otros modelos similares pueden configurarse en los sistemas de filtrado para identificar patrones de comandos de sistema, inyecciones, *scripts* maliciosos o intentos de evasión de seguridad en las salidas del modelo.

[15] https://ai.meta.com/research/publications/llama-guard-llm-based-input-output-safeguard-for-human-ai-conversations/

05.6. Agencia Excesiva

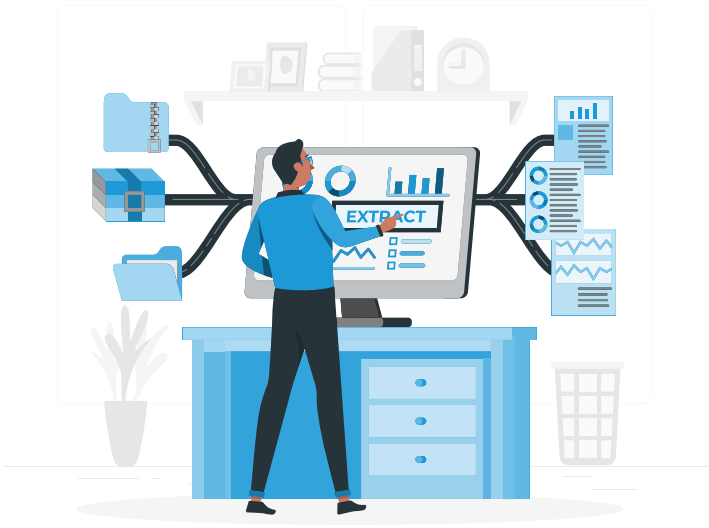
Esta vulnerabilidad se refiere a escenarios donde un sistema de **IA** posee capacidades que sobrepasan las necesarias para su función prevista, creando así oportunidades para acciones potencialmente maliciosas o dañinas. Esto puede ser debido a tres causas principales:

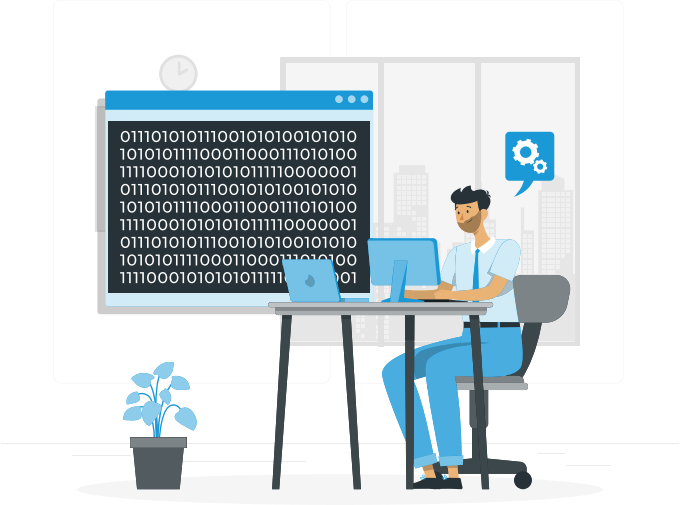
01 | La **funcionalidad excesiva** introduce un riesgo cuando un sistema de **IA** tiene acceso a más herramientas, **APIs** o capacidades de las estrictamente necesarias para cumplir su propósito.

02 | Los **permisos excesivos** representan una vulnerabilidad cuando los sistemas de **IA** operan con privilegios elevados en relación con sus requisitos funcionales. Un ejemplo de esta vulnerabilidad se manifiesta en sistemas que utilizan privilegios elevados para conectarse a bases de datos o para ejecutarse, permitiéndoles realizar acciones con permisos superiores a los necesarios.

03 | La **autonomía excesiva** se manifiesta cuando los sistemas de **IA** pueden ejecutar acciones sin mecanismos de verificación, aprobación o supervisión. Las implementaciones de sistemas completamente autónomos sin mecanismos de verificación apropiados pueden introducir este tipo de vulnerabilidades.

Para mitigar esta vulnerabilidad se recomienda la aplicación del principio de privilegio: cada sistema o componente de **IA** es necesario que opere con los privilegios y accesos mínimos necesarios para cumplir su función específica.





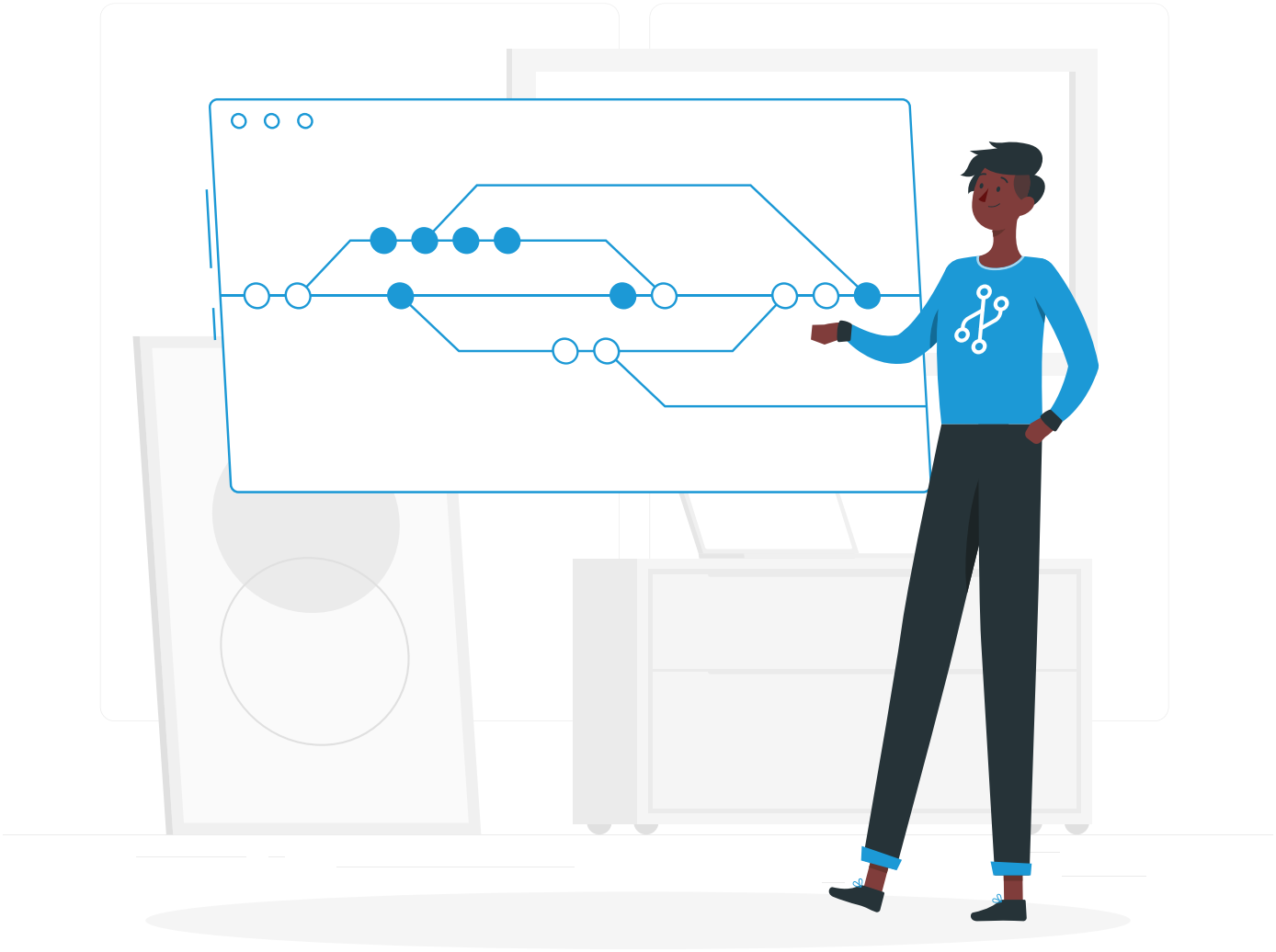
05.7. Filtración del *prompt* del sistema

Esta vulnerabilidad se produce cuando un usuario malintencionado logra que el modelo revele sus instrucciones originales o ***prompts*** del sistema que deberían permanecer ocultos. Es una variación de la vulnerabilidad de inyección de ***prompt*** donde el atacante puede extraer información confidencial del ***backend*** del sistema.

Como ya se ha comentado anteriormente OpenWeb UI ofrece un sistema llamado ***Pipelines*** que actúa contra la filtración de ***prompts*** del sistema. Esta característica funciona como un marco de ***plugins***, permitiendo añadir capacidades para interceptar y procesar los ***prompts*** de usuario antes de que lleguen al modelo **LLM**. A través de sus componentes principales es posible examinar, modificar o rechazar ***prompts*** potencialmente maliciosos.

Un plugin muy utilizado es el ***Prompt Enhancer Filter*** desarrollado por la comunidad, que reformula los ***prompts*** entrantes siguiendo buenas prácticas de seguridad, eliminando intentos de manipulación del sistema e implementando restricciones que dificultan los ataques de inyección.

Complementando el sistema de ***Pipelines***, ***OpenWeb UI*** implementa estrategias adicionales de mitigación como la segregación de contexto, que mantiene la información sensible aislada de las interacciones directas con el usuario. Además de integrar técnicas generales como el filtrado de salidas para detectar y bloquear intentos de filtración de ***prompts***, creando así un enfoque de seguridad en capas que reduce significativamente el riesgo de exposición de instrucciones del sistema.



05.8. Debilidades en Vectores y *Embeddings*

Esta vulnerabilidad se produce por la dependencia de los sistemas de IA de representaciones vectoriales para procesar datos matemáticamente, lo que puede permitir a actores malintencionados explotar estas representaciones para extraer información sensible.

Un ejemplo puede ser el ataque de inversión de *embeddings*, donde los atacantes pueden reconstruir datos originales a partir de su representación vectorizada, provocando filtraciones, envenenamiento de datos o manipulación del comportamiento del modelo. Esta vulnerabilidad afecta especialmente a sistemas que utilizan bases de conocimiento vectorizadas para proporcionar respuestas.

OpenWeb UI ofrece diversas herramientas para mitigar estos riesgos permitiendo seleccionar entre opciones como *chroma*, *milvus*, *qdrant*, *opensearch* y *pgvector* según los requisitos de seguridad que se establezcan.

Además, la plataforma implementa motores de *embedding* configurables a través de la variable *RAG_EMBEDDING_ENGINE*, permitiendo elegir entre *SentenceTransformers*, API de Ollama o API de OpenAI, lo que facilita la implementación de motores con características de seguridad mejoradas.

05.9. Desinformación

La vulnerabilidad de desinformación se produce cuando un **LLM** genera información falsa, engañosa y aparentemente creíble.

Este fenómeno, también conocido como *alucinaciones* del modelo, representa un riesgo debido a su alto impacto reputacional y la relativa facilidad con que puede ocurrir.

Para reducir la aparición de alucinaciones, la plataforma permite la integración de bases de conocimiento que el modelo puede consultar antes de generar respuestas.

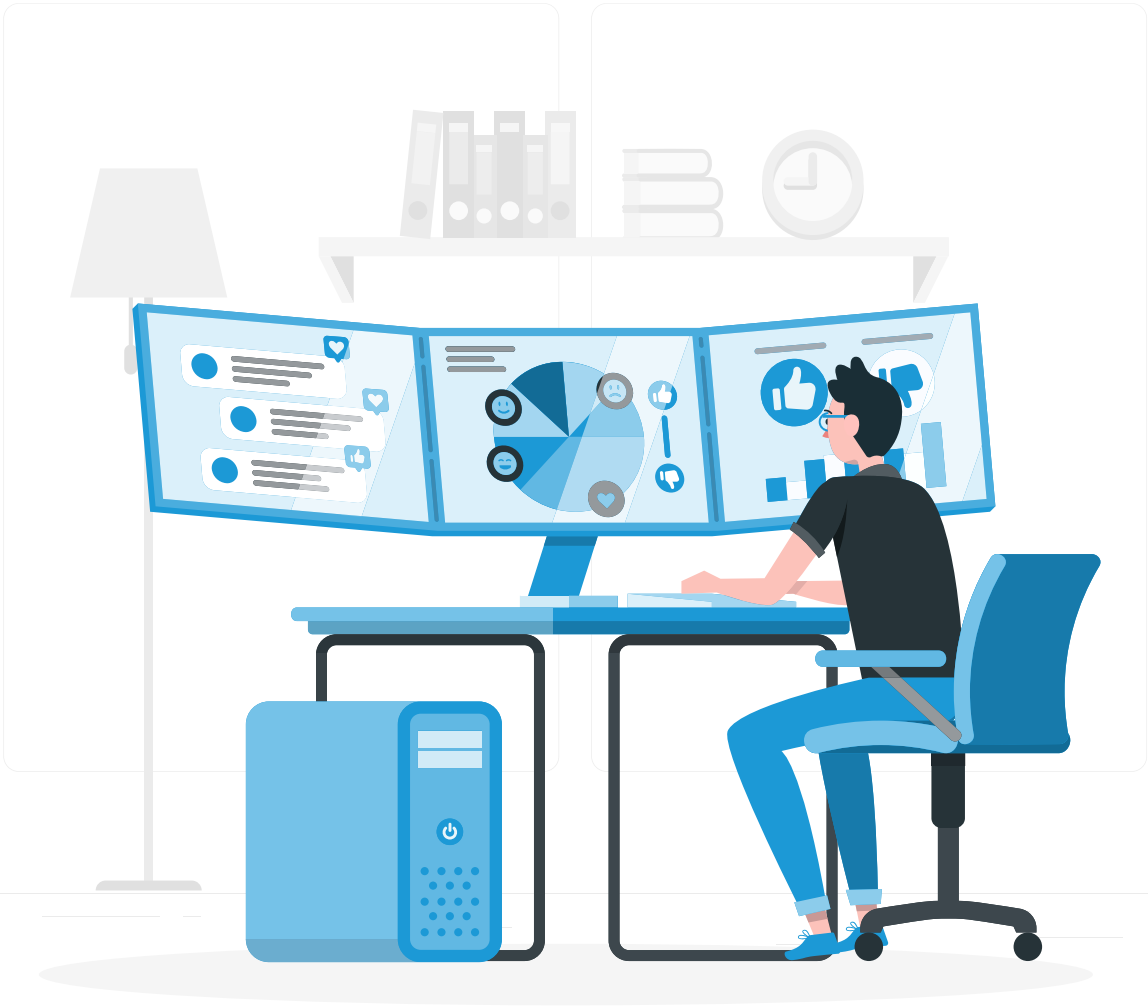
Es posible personalizar los parámetros avanzados del modelo para reducir las alucinaciones. Como recomendación general se pueden establecer los siguientes valores, aunque será necesario evaluar todos los parámetros teniendo en cuenta la aplicación práctica de cada modelo en particular.

Ajustando el parámetro *temperatura* entre los valores (*0.2-0.4*) hará que el modelo sea más determinista al generar los resultados.

Si se establece el parámetro *top-p* en valores menores a *0.7* y *top-k* entre valores *40-100*, será posible ajustar la creatividad del modelo.

También es posible limitar la longitud máxima de las respuestas, en este caso estableciendo valores entre el rango (*256-512*) *tokens*.

Adicionalmente, se recomienda establecer un *system prompt* que delimite las acciones que puede realizar o no el modelo.



05.10. Consumo de recursos excesivo

Esta vulnerabilidad se produce cuando un servicio **LLM** permite a los usuarios realizar inferencias excesivas y sin control adecuado pudiendo provocar una denegación o degradación del servicio.

OpenWeb UI puede configurarse para implementar límites de uso por usuario, restringiendo el número de solicitudes o *tokens* generados en un período determinado. Esto evita que usuarios individuales consuman una cantidad excesiva de recursos computacionales.

06.

ANEXO: GLOSARIO DE TÉRMINOS

IA (Inteligencia Artificial): rama de la informática que desarrolla sistemas capaces de realizar tareas que normalmente requieren inteligencia humana, como el reconocimiento de voz, la toma de decisiones y el procesamiento del lenguaje natural.

Prompt: instrucción o entrada proporcionada a un sistema de IA, especialmente a modelos de lenguaje, para generar una respuesta específica.

Chatbot: programa informático diseñado para simular conversaciones con usuarios humanos mediante texto o voz, utilizando reglas predefinidas o inteligencia artificial.

SBOM (Software Bill of Materials): lista detallada de los componentes de software incluidos en un producto, utilizada para mejorar la seguridad y la gestión de dependencias.

LLM (Large Language Model): modelo de lenguaje de gran tamaño basado en redes neuronales, entrenado con grandes volúmenes de texto para comprender y generar lenguaje natural de manera avanzada.

Token: unidad mínima de procesamiento en un modelo de lenguaje, que puede ser una palabra, parte de una palabra o incluso un carácter según la segmentación utilizada.

Contenedor Docker: entorno de ejecución ligero y portátil que encapsula aplicaciones y sus dependencias, permitiendo su despliegue en diferentes sistemas sin conflictos.

Imagen Docker: plantilla que contiene todo lo necesario para ejecutar un contenedor, incluyendo el código de la aplicación, bibliotecas y configuraciones.

Git: sistema de control de versiones que permite a los desarrolladores gestionar el historial de cambios en el código fuente de un proyecto.

Backend: parte de una aplicación o sistema que maneja la lógica del negocio, el procesamiento de datos y la interacción con la base de datos, operando en el servidor.

Plugin: módulo o extensión que agrega funcionalidades adicionales a un software o plataforma.

GPU (Unidad de Procesamiento Gráfico): procesador especializado en el cálculo paralelo, optimizado para el procesamiento de gráficos y tareas de alto rendimiento como la inteligencia artificial y la minería de datos.

Ollama: plataforma o software relacionado con la ejecución y gestión de modelos de lenguaje avanzados de manera local o en la nube.

GOBIERNO DE ESPAÑA
MINISTERIO DE DEFENSA

CCN
centro criptológico nacional

MARZO 2025

Instalación y Securización de un **ChatBot** con LLM de forma local.