MÁSTER UNIVERSITARIO EN ENXEÑARÍA INFORMÁTICA
Mención de Máster Universitario Excelente da Xunta de Galicia

UNIVERSIDADE DA CORUÑA

Facultade de Informática

Máster Universitario en Enxeñaría Informática

TRABALLO FIN DE MÁSTER

guardIAn: chatbot baseado en LLM's especializado en ciberseguridade.

Estudiante: Juan Toirán Freire

Dirección: Eliseo Bao Souto

Miguel Anxo Pérez Vila

A Coruña, junio de 2025.

Dedicatoria

Agradecimientos

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Resumen

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Abstract

Hello, here is some text without a meaning. This text should show what a printed text will look like at this place. If you read this text, you will get no information. Really? Is there no information? Is there a difference between this text and some nonsense like "Huardest gefburn"? Kjift – not at all! A blind text like this gives you information about the selected font, how the letters are written and an impression of the look. This text should contain all letters of the alphabet and it should be written in of the original language. There is no need for special content, but the length of words should match the language.

Palabras clave:

- First itemtext
- Second itemtext
- Last itemtext

Keywords:

- First itemtext
- Second itemtext
- Last itemtext

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Neste primeiro capítulo farase unha contextualización do proxecto desenvolto co fin de poder dar a entender ao lector de que tratará este traballo, que o motiva e ser unha guía proceder para a lectura do mesmo.

IMPORTANTE, POR SE ALGUÉN LEE ESTO POR AGORA. AÍNDA NON ENGADÍN NINGUNHA FOTO, ESQUEMA NIN OUTRO TIPO DE IMAXE PORQUE ESTOUME CENTRANDO EN ESCRIBIR O MÁXIMO QUE PODA, IGUAL QUE AÍNDA NON FIXEN REFERENCIA CORRECTAMENTE A BILBIOGRAFÍA, NIN ÁS SIGLAS, NI AOS APARTADOS/CAPÍTULOS

1.1 Motivación do proxecto

A motivación principal que deu orixe ao desenvolvemento deste proxecto é a crecente importancia da ciberseguridade tanto no ámbito empresarial como no persoal. Vivimos nun contexto altamente dixitalizado, onde practicamente todas as actividades cotiás como comercio, comunicación, administración, educación ou sanidade dependen de sistemas informáticos interconectados. Esta dependencia expón os sistemas a un número crecente de ameazas: ataques de denegación de servizo, ransomware, roubo de datos ou explotación de vulnerabilidades entre outras.

As organizacións, independentemente do seu tamaño, enfróntanse ao reto de protexer os seus activos dixitais. Isto require recursos, tempo e, sobre todo, persoal altamente cualificado. Un dos principais desafíos que atopan os equipos de seguridade informática (blue team) é a análise de grandes volumes de datos, que se volven extremadamente difíciles de manexar de forma eficiente cando se produce un incidente. Tras un ataque, analizar o que pasou pode supoñer horas ou incluso días de revisión manual de rexistros, dificultando unha resposta áxil.

Paralelamente, a irrupción dos modelos de linguaxe de gran tamaño (LLMs, do inglés Large Language Models) abre un novo horizonte de posibilidades. Estes modelos son capaces

de procesar e xerar linguaxe natural con precisión, e xa están sendo utilizados en numerosos ámbitos, como atención ao cliente, programación ou educación. Non obstante, o seu uso no dominio da ciberseguridade aínda está en fase incipiente, especialmente no ámbito open-source, o cal representa unha oportunidade única de innovación e impacto real.

Ademais, os LLM tamén introducen novos riscos: poden ser utilizados por persoas sen coñecementos técnicos para realizar accións malintencionadas. Esta realidade acentúa aínda máis a necesidade de contar con ferramentas éticas, seguras e especializadas que poidan actuar como apoio para profesionais da seguridade.

Neste contexto nace guardIAn, un proxecto que ten como obxectivo desenvolver un chatbot baseado en LLM especializado en tarefas de ciberseguridade, que poida actuar como asistente para blue teams e red teams. Esta ferramenta pretende ofrecer recomendacións, analizar logs, suxerir probas de penetración e contribuír á mellora continua da seguridade das organizacións. Todo isto apoiado nun enfoque de Retrieval-Augmented Generation (RAG) que permite acceder a información específica e actualizada procedente de fontes expertas, como foros especializados.

Outro aspecto clave da motivación é a necesidade de democratizar o acceso a este tipo de tecnoloxía. Ao ser un proxecto open-source, guardIAn poderá ser utilizado, modificado e mellorado pola comunidade, contribuíndo a un ecosistema de seguridade máis colaborativo e accesible. Isto encaixa tamén coa filosofía de aprendizaxe continua e código libre que caracteriza ao ámbito universitario e á comunidade de ciberseguridade.

Finalmente, este proxecto representa unha oportunidade persoal e profesional para poñer en práctica coñecementos adquiridos durante o máster, explorar tecnoloxías punteiras e contribuír, dentro das súas posibilidades, a resolver un problema real cun impacto potencial considerable.

1.2 Contexto e problemática da ciberseguridade

A ciberseguridade converteuse nunha preocupación estratéxica de primeira orde na sociedade actual. O avance imparable da dixitalización supón que cada vez máis datos persoais, corporativos e institucionais se almacenan e transmiten a través de redes interconectadas. Este escenario, aínda que ofrece enormes vantaxes a nivel económico e social, tamén multiplica os vectores de ataque e vulnerabilidades que poden ser explotadas por axentes malintencionados.

Nun contexto no que calquera dispositivo conectado pode ser unha porta de entrada a un sistema máis complexo, os ataques informáticos están a ser máis frecuentes, sofisticados

e difíciles de detectar. As técnicas de intrusión evolucionan constantemente, e os ciberdelincuentes dispoñen de ferramentas cada vez máis potentes, moitas veces impulsadas pola mesma intelixencia artificial que tamén usan os defensores. Isto crea unha carreira armamentística tecnolóxica entre atacantes (red team) e defensores (blue team), que obriga a estar en constante actualización.

A día de hoxe, moitas empresas e administracións públicas enfróntanse a múltiples retos no eido da seguridade informática:

- **Falta de persoal cualificado** con competencias específicas en análise forense, detección de ameazas ou resposta ante incidentes.
- **Volume masivo de datos** xerados por sistemas, redes e aplicacións, que complican a identificación de patróns anómalos.
- **Tempo de reacción limitado,** o que implica que, se non se detecta un ataque a tempo, os danos poden ser irreversibles.
- **Riscos legais e reputacionais,** derivados do incumprimento de normativas como o RGPD, ou da exposición pública de brechas de seguridade.

Ao mesmo tempo, os cidadáns tamén se senten cada vez máis vulnerables, pois existen riscos reais sobre os seus datos bancarios, redes sociais ou servizos na nube. A protección da información persoal xa non é unha cuestión técnica, senón tamén social e ética.

Neste marco, o papel da automatización e da intelixencia artificial emerxe como un aliado clave. Tecnoloxías como os LLMs (Large Language Models) permiten tratar a linguaxe natural con gran precisión, abrindo a posibilidade de crear asistentes que poidan analizar rexistros (logs), explicar problemas de seguridade, suxerir accións e mesmo realizar auditorías básicas de forma asistida.

Non obstante, o uso de modelos avanzados tamén pode ser perigoso se cae en mans inadecuadas. Un usuario sen coñecementos técnicos podería, por exemplo, empregar un LLM para identificar vulnerabilidades e explotalas, xerando así un novo tipo de ameaza. Isto reforza a necesidade de deseñar sistemas seguros, éticos e transparentes, que axuden aos profesionais a defender, e non a atacar.

A problemática real radica en que a análise e resposta ante incidentes segue sendo, en moitos casos, manual e intensiva en tempo e recursos, algo que xa non é sustentable na escala actual. Aquí é onde entra a proposta de guardIAn, un chatbot que combina as capacidades dun modelo de linguaxe cun sistema de recuperación de información relevante (RAG), optimizado

para reducir tempos de resposta, detectar ameazas e xerar recomendacións baseadas en coñecemento actualizado. Neste contexto global de crecente complexidade e risco, o presente proxecto non só é tecnoloxicamente pertinente, senón tamén socialmente necesario.

1.3 Obxectivos xerais e específicos

Este Traballo de Fin de Máster ten como obxectivo principal o deseño, desenvolvemento e avaliación dun chatbot baseado en modelos de linguaxe de gran tamaño (LLMs) especializado no ámbito da ciberseguridade. Trátase dunha ferramenta que, mediante o uso de tecnoloxías avanzadas de intelixencia artificial, busca optimizar tarefas críticas como a análise de logs, auditorías de seguridade e probas de penetración, proporcionando unha resposta automatizada, contextual e fundamentada en información actualizada. O problema que se aborda é concreto e relevante: en situacións reais de ataque ou sospeita de actividade maliciosa, os equipos de ciberseguridade perden unha gran cantidade de tempo revisando rexistros e investigando a orixe e natureza da ameaza. A complexidade destes procesos, xunto co elevado volume de datos xerados polos sistemas informáticos, fai que a resposta sexa lenta e, ás veces, ineficiente. A través deste traballo, preténdese reducir ese tempo de resposta e aumentar a precisión das análises mediante o soporte dunha ferramenta intelixente, áxil e de código aberto.

Este chatbot está pensado como un complemento para profesionais de ciberseguridade, tanto desde o enfoque do blue team (defensa e resposta ante incidentes) como do red team (avaliación de vulnerabilidades e simulacións de ataque). A ferramenta non substitúe ao experto humano, senón que serve de apoio técnico para axilizar tarefas rutineiras ou preliminares, favorecendo unha toma de decisións máis informada. Así mesmo, a interface web que se desenvolverá permitirá interactuar co chatbot dunha maneira sinxela e efectiva, democratizando o seu uso e garantindo unha mellor accesibilidade para o público profesional.

A continuación detállanse os obxectivos específicos que se perseguen durante o desenvolvemento do proxecto:

- **Deseñar e implementar un chatbot open-source especializado en ciberseguridade,** capaz de realizar tarefas como análise de rexistros, suxestión de melloras na configuración de sistemas ou execución de consultas sobre vulnerabilidades coñecidas.
- **Recolectar e procesar un conxunto de datos diverso e de calidade** a partir de fontes abertas, especialmente foros especializados en seguridade informática como Gentoo. Estes datos serán utilizados para axustar o comportamento do modelo de linguaxe mediante técnicas de fine-tuning.
- **Aplicar técnicas de adaptación eficiente como LoRA (Low-Rank Adaptation)**

sobre o modelo de linguaxe LLaMA 3.1, para especializalo nas tarefas concretas do dominio de ciberseguridade con consumo reducido de recursos.

- **Construír índices densos (embeddings)** co contido dos foros e logs para permitir a busca semántica eficiente, utilizando ElasticSearch como motor de indexado.
- **Integrar o modelo e os índices mediante unha pipeline de Retrieval-Augmented Generation (RAG)** empregando o framework LangChain, para combinar a xeración lingüística cunha recuperación precisa de información relevante.
- **Desenvolver unha API robusta e funcional con FastAPI,** que permita a comunicación entre a interface web e o modelo de linguaxe, asegurando un fluxo de datos consistente e controlado.
- **Construír unha interface web moderna e funcional utilizando o framework Django,** ofrecendo un punto de acceso centralizado ao sistema e mellorando a experiencia de usuario.
- **Garantir a viabilidade técnica e económica do proxecto,** facendo uso de software open-source, recursos xa dispoñibles no grupo de investigación, e estimacións realistas de custos asociados.
- **Documentar o proceso de desenvolvemento,** aplicando metodoloxía áxil (Scrum adaptado), con control de versións, planificación por sprints e seguimento continuo das tarefas.

O público destinatario principal deste desenvolvemento inclúe profesionais da ciberseguridade, analistas SOC (Security Operations Center), pentesters, membros de equipos de resposta a incidentes e, en xeral, calquera persoa con coñecementos técnicos que precise automatizar ou apoiar a súa labor de análise e defensa informática. Ao ser open-source, a ferramenta tamén poderá ser adoptada e mellorada por centros de formación, investigadores ou pequenas empresas sen grandes recursos, fomentando unha cultura de seguridade máis accesible.

1.4 Público obxectivo

O público obxectivo deste proxecto está claramente definido e responde ás necesidades actuais do sector da ciberseguridade. guardIAn, como chatbot especializado, está deseñado para ser unha ferramenta de apoio técnico que permita optimizar o traballo de profesionais e axentes implicados na protección e avaliación da seguridade informática, tanto a nivel organizativo como operativo.

1.4.1 Profesionais da ciberseguridade (Blue Team / Red Team)

O principal grupo destinatario son os profesionais de ciberseguridade, en especial os pertencentes a equipos blue team e red team. Os primeiros encárganse da defensa dos sistemas informáticos dunha organización, centrándose na detección, monitorización e mitigación de ameazas, mentres que os segundos simulan ataques reais para identificar vulnerabilidades e avaliar a resiliencia da infraestrutura.

Para este tipo de perfís, guardIAn pode ofrecer un apoio valioso en tarefas como:

- Análise e explicacion de erros cometidos na creación de infraestruturas ou sistemas.
- Xeración de recomendacións técnicas para reforzar a seguridade.
- Guía interactiva na execución de probas de penetración ou recolección de evidencias.
- Resumo de ameazas coñecidas ou recentes, baseadas en fontes actualizadas do sector.

1.4.2 Pequenas e medianas empresas (PEMEs) sen equipos especializados

Outro público importante son as organizacións de pequeno ou mediano tamaño que non dispoñen dun equipo propio de ciberseguridade. Estas entidades adoitan ter dificultades para asumir os custos asociados a servizos profesionais ou solucións comerciais complexas. A posibilidade de empregar un chatbot open-source, accesible a través dunha interface web moderna, supón unha oportunidade de mellorar a súa postura en seguridade cun investimento moi reducido.

Para este segmento, guardIAn pode servir como:

- Primeira liña de asesoramento automatizado ante dúbidas ou alertas.
- Ferramenta de formación interna para persoal técnico con responsabilidades compartidas.
- Apoio para revisión de configuracións, rexistros e recomendacións de bo uso.

1.4.3 Profesionais en formación e estudantes de ciberseguridade

O uso de tecnoloxía aberta e documentada con fins educativos tamén permite que o proxecto sexa útil para persoas en proceso de aprendizaxe no eido da ciberseguridade, xa sexa en graos, mestrados ou cursos especializados. Ao empregar a ferramenta, poden explorar:

- Casos reais de análise forense simulada.

- Interacción directa cun sistema que ofrece respostas técnicas con base en datos reais.
- A estrutura interna dun chatbot de seguridade, para entender o seu deseño e arquitectura.

Así mesmo, o código aberto permite aos estudantes modificar e experimentar co proxecto, facilitando o seu uso como base para prácticas, TFGs ou investigacións complementarias.

1.4.4 Comunidade open-source e investigadores en IA aplicada á seguridade

O carácter open-source do proxecto abre tamén o seu uso á comunidade de desenvolvedores e investigadores que traballan na intersección entre IA e ciberseguridade. Estes perfís poden estar interesados en:

- Explorar o enfoque de Retrieval-Augmented Generation aplicado a seguridade informática.
- Ampliar a base de datos do sistema ou mellorar o seu rendemento.
- Integrar guardIAn noutras ferramentas ou sistemas existentes.

Este público poderá beneficiarse especialmente da modularidade do sistema, da documentación dispoñible e da posibilidade de contribuír ao seu desenvolvemento ou reutilizar compoñentes en proxectos propios.

1.5 Estrutura da memoria

Para poder entender de forma máis sinxela a memoria vamos a mostrar a continuación un listado explicando a rasgos xerais cada capítulo.

- Introdución: Expón a motivación do traballo, o panorama e os retos actuais da ciberseguridade. Define obxectivos xerais/específicos, públicos destinatarios e a estrutura da memoria.
- Fundamentos e conceptos clave: Revisa as bases técnicas: web crawlers, LLMs, LoRA, RAG, embeddings e filosofía open-source. Establece o marco teórico necesario para entender o resto do traballo.
- Tecnoloxías e ferramentas empregadas: Enumera linguaxes (Python, JS), frameworks (LangChain, FastAPI, Django, React) e utilidades (Scrapy, GitHub…). Xustifica a elección para construír o sistema.

- Planificación e metodoloxía: Descríbese unha versión adaptada de Scrum con sprints, recursos, custos estimados e xestión de riscos. A metodoloxía garante iteracións rápidas e control de calidade.
- Extracción de datos: Explica fontes e deseño do crawler en Scrapy, políticas de scraping e limpeza/anotación de logs. Detalla o almacenamento e indexación en Elasticsearch.
- Creación da web: Presenta a arquitectura con frontend en React, backend Django/FastAPI e fluxo RAG con LangChain+Elasticsearch. Mostra como se integra todo para ofrecer unha experiencia fluída ao usuario.
- Entrenamento do modelo e xeración de embeddings: Selecciona Llama 3.1, aplica LoRA para o fine-tuning e xera embeddings. Conecta o modelo á pipeline RAG para respostas contextualizadas en ciberseguridade.
- Desenvolvemento do proxecto: Recolle requisitos e historias de usuario e describe a execución práctica dos sprints. Evidencia o avance ata obter unha versión funcional.
- Resultado do desenvolvemento: Presenta a aplicación operativa, código e datasets liberados como open-source. Resume a utilidade alcanzada e a dispoñibilidade dos recursos.
- Conclusións, desenvolvementos futuros e relación co MUEI: Sintetiza os logros, aprendizaxes e impacto académico. Propón melloras futuras e vincula o traballo coas materias do máster.

Capítulo 2

Fundamentos e conceptos clave

2.1 Web Crawler

Un web crawler (ou araña da web) é un programa automatizado que percorre sistematicamente páxinas web para recolectar e indexar o seu contido. Este tipo de ferramentas xurdiron a mediados dos anos 90 co propósito de alimentar os primeiros motores de busca (como WebCrawler, AltaVista ou Google) e dende entón convertéronse en compoñentes esenciais para a recuperación de información na web. En esencia, un rastrexador web desprazase a través da rede seguindo hipervínculos: comeza cunha lista inicial de URL (chamada sementes) e, por cada páxina visitada, extrae as ligazóns contidas nela e engádeas á cola de pendentes para visitar a continuación. Deste xeito, constrúese de forma iterativa un índice masivo de páxinas web, o que permite que os motores de busca ofrezan resultados rápidos e relevantes aos usuarios. Ademais da indexación para buscadores, os crawlers teñen outras aplicacións, como a detección de ligazóns rotas, a monitorización de cambios en sitios web ou a recolección de datos específicos (prezos de produtos, publicacións en redes sociais, etc.). Este comportamento metódico valeulle aos crawlers o alcume de "arañas" da web, en alusión a como tecen unha rede percorrendo fío a fío (ligazón a ligazón) o World Wide Web.

Arquitectura e funcionamento interno: Internamente, un web crawler consta de varios módulos clave: un xestor de URL pendentes (tamén chamado fronteira de rastrexo), un módulo de descarga (para facer as peticións HTTP e obter o HTML de cada páxina) e un parser (que analiza o HTML extraendo novo URLs).

Este algoritmo elemental impleméntase, na práctica, dun xeito distribuído e altamente optimizado para poder escalar a miles de millóns de páxinas. Por exemplo, os crawlers profesionais implementan políticas de prioridade (p.ex. rastrexo Breadth-First vs. Depth-First), regras de cortesía (respectando o ficheiro robots.txt e evitando sobrecargar servidores cun retardo entre accesos) e mecanismos de duplicación para non visitar dúas veces a mesma URL ou páxinas con contido idéntico. Alén diso, a arquitectura adoita ser multifío (multi-threaded)

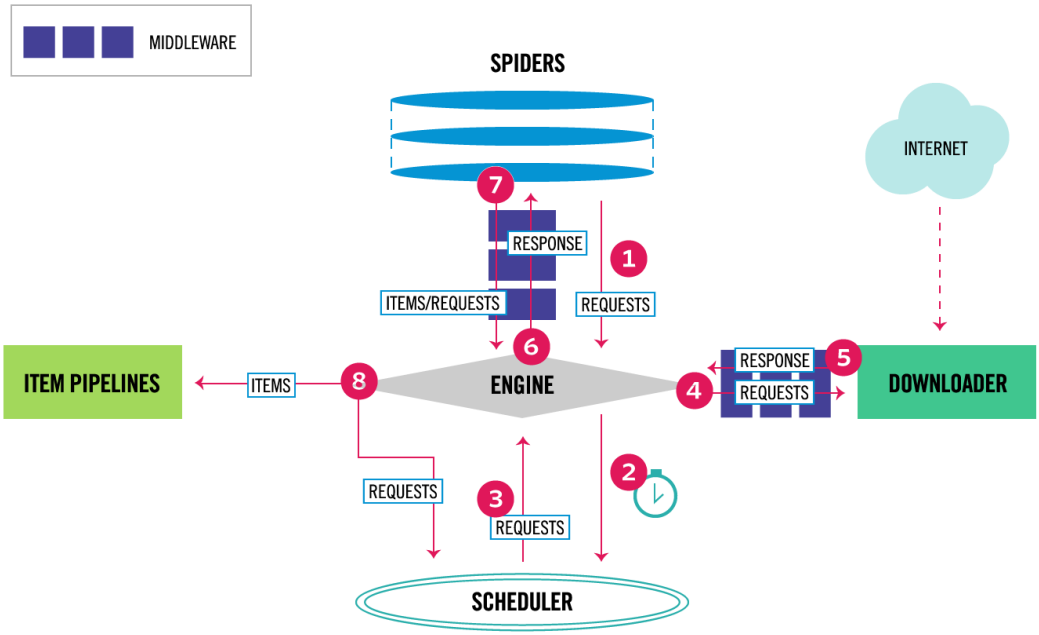

Figura 2.1: Esquema funcionamiento crawler.

ou incluso distribuída en múltiples máquinas, o que permite rastrexar diferentes sitios en paralelo aproveitando a banda larga de Internet. Estas optimizacións foron necesarias a medida que o tamaño da web medraba exponencialmente: os primeiros crawlers, a principios dos 90, podían indexar unhas cantas milleiras de páxinas, mentres que hoxe en día Googlebot e outras arañas comerciais rastrexan decenas de miles de millóns de URLs ao mes.

Vantaxes e limitacións: A principal vantaxe dos web crawlers é que automatizan a recolección masiva de información dispoñible publicamente. Isto fai posible construír índices moi completos da web e manter actualizados os motores de busca co novo contido que xorde cada día. Así mesmo, en contextos específicos, un crawler pode axilizar tarefas de open-source intelligence (OSINT) en ciberseguridade, como rastrexar repositorios de malware coñecidos ou enumerar páxinas dun dominio en busca de vulnerabilidades. Porén, existen importantes limitacións:

- **Cobertura e eficiencia:** É inalcanzable rastrexar toda a web, polo que hai que equilibrar profundidade de exploración e amplitude de sitios cubertos.
- **Custos computacionais:** Un crawler a gran escala require moito ancho de banda, procesamento e almacenamento.

- **Riscos legais e éticos:** É preciso respectar as directrices robots.txt e termos de uso dos sitios; un rastreo agresivo pode ser considerado DoS ou intrusión.
- **Contido dinámico e APIs pechadas:** Moita información hoxe está xerada dinámicamente ou detrás de interfaces de API e non é accesible para un crawler tradicional.

Relación co proxecto: No contexto deste TFM, o web crawler empregase para recoller automaticamente información relevante que alimente a base de coñecemento do chatbot. Rastrexando foros especializados en ciberseguridade, como os de Debian e Gentoo, para extraer contidos que logo sexan indexados. Deste xeito, garántese que o chatbot dispoña de información actualizada sobre novas ameazas ou discusións técnicas sen depender exclusivamente do coñecemento almacenado no modelo de linguaxe. A integración dun módulo de crawling neste proxecto open-source facilita que o asistente poida expandir o seu alcance de respostas empregando datos reais da web auditada (por exemplo, rastrexando o propio foro dunha comunidade en busca de exposicións ou referencias a prácticas de seguridade). Aínda que o núcleo do chatbot é un modelo de IA, os web crawlers actúan como recolectores para alimentalo con datos mellorando a calidade e actualidade das respostas.

2.2 Modelos de linguaxe de gran tamaño (LLMs)

Un modelo de linguaxe é un sistema de intelixencia artificial capaz de estimar a probabilidade das secuencias lingüísticas e de xerar texto en linguaxe natural. Máis formalmente, dado un contexto, o modelo asigna unha probabilidade a cada posible seguinte token, permitindo predicir palabras ou símbolos para compoñer frases. Os Modelos de Linguaxe de Gran Tamaño – ou LLMs, polas súas siglas en inglés – son aqueles que teñen milleiros de millóns ou incluso billóns de parámetros adestrables, e que foron adestrados con cantidades masivas de datos textuais. Este incremento de escala extraordinario fronte aos modelos de linguaxe clásicos (por exemplo, os modelos n-gram ou os primeiros RNNs dos anos 90 e 2000) demostrou un salto cualitativo nas capacidades do modelo: os LLM modernos poden comprender consultas complexas, traducir entre múltiples idiomas, resumir documentos extensos e mesmo xerar respostas creativas e con coherencia semellante á humana. Todo isto é posible grazas aos milleiros de millóns de parámetros que conteñen, os cales lles permiten capturar patróns lingüísticos altamente complexos e matices semánticos que van máis aló do alcance de modelos máis pequenos.

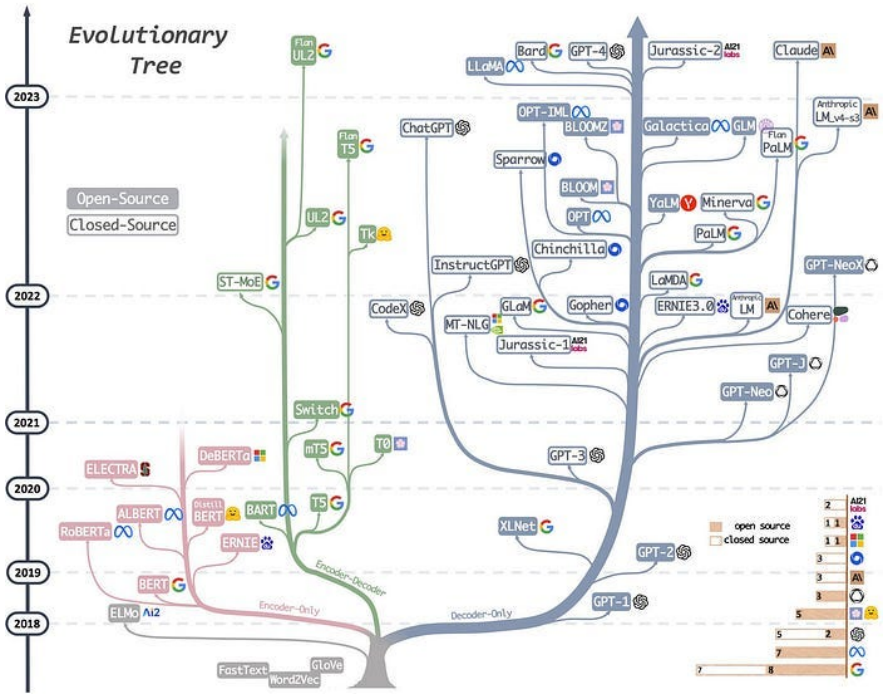

Figura 2.2: Evolución dos LLMs actuais.

Arquitectura e evolución: Os LLMs actuais baséanse tipicamente na arquitectura Transformer, caracterizada por un mecanismo de atención que avalía a relevancia recíproca entre todas as palabras dunha secuencia. Este deseño permitiu superar limitacións das redes neuronais recurrentes e LSTM anteriores, logrando entrenar modelos con centos de capas en paralelo e cun contexto moi amplo. A combinación do Transformer con enormes corpus de texto (orde de terabytes, incluíndo Wikipedia, libros, artigos e a web) deu lugar a modelos fundacionais de propósito xeral. Algúns fitos destacados na evolución dos LLMs son BERT, autoencoders bidireccionais que capturou relacións contextuais en textos curtos; GPT-3 que o popularizou e os modelos da familia LLaMA de Meta AI, que demostraron que é posible acadar rendemento competitivo con menos parámetros mediante un adestramento eficiente. En xeral, un LLM consiste nunha capa de embedding que transforma cada token de entrada nun vector continuo, seguida por ducias de capas Transformer (bloques de autoatención multi-cabeza e feed-forward) e finalmente unha capa de saída que devolve unha distribución de probabilidade sobre o seguinte token. Durante o adestramento, axústanse millóns de pesos internos minimizando a perda de entropía cruzada entre as predicións do modelo e os tokens reais nos datos de adestramento. Esta fase de pre-adestramento xeral pode complementarse despois con técnicas como fine-tuning supervisado ou aprendizaxe por reforzo con feedback humano (RLHF) para especializar o modelo en certas tarefas.

Os LLMs destacan por manexar un coñecemento enciclopédico implícito e por xerar texto fluído, coherente e gramaticalmente correcto, o que lles permite redactar explicacións técnicas, escribir código ou manter diálogos naturais. A súa maior fortaleza é a versatilidade: co prompting axeitado ou cun pequeno fine-tuning poden traducir, resumir, responder preguntas ou xerar contido sen necesidade de arquitecturas específicas para cada tarefa. Como modelos fundacionais, serven ademais de base para especializacións posteriores con poucos datos adicionais.

Con todo, presentan limitacións importantes. Alucinan con facilidade —producen respostas plausibles pero incorrectas— porque só modelan probabilidades lingüísticas, non a realidade factual. Tamén poden perpetuar sesgos presentes nos datos de adestramento. A nivel operativo, requiren grandes recursos computacionais tanto para adestrar coma para inferir, o que dificulta o seu uso en contornas con hardware limitado. Por último, o seu coñecemento é estático: descoñecen feitos posteriores á data de corte do corpus a non ser que se complemente o modelo cun mecanismo externo de actualización, como unha pipeline RAG.

Relación co proxecto: O chatbot de auditoría e ciberseguridade desenvolto neste TFM está impulsado no seu núcleo por un LLM. En concreto, utilizouse un modelo LLaMA 3.1 de Meta, pertencente á categoría dos LLM open-source de última xeración, afinado posteriormente para o noso dominio (véxase sección TÑO QUE BUSCAR A SECCIÓN CORRESPONDENTE) co fin de especializalo en auditorías de ciberseguridade. Este modelo de linguaxe de gran tamaño encárgase de xerar as respostas ás preguntas do usuario explicando conceptos técnicos, normativas, boas prácticas, etc., dun xeito conversacional. A elección dun LLM permite que o chatbot teña unha comprensión ampla da linguaxe natural dos auditores, poida interpretar preguntas formuladas de múltiples maneiras e elaborar respostas detalladas aproveitando o coñecemento almacenado nos seus parámetros. Ademais, ao estar reforzado cunha pipeline RAG (Retrieval-Augmented Generation) e co fine-tuning específico, o LLM do chatbot combina as capacidades xenéricas (aprendidas de grandes corpus públicos) coa información específica e actualizada do dominio de ciberseguridade. En resumo, o LLM é o "cerebro" do asistente: un modelo xeralista potente (LLaMA) que, mediante adaptación, se converte nun experto virtual capaz de asistir en tarefas de auditoría de seguridade informática.

2.3 Fine-tuning e adaptación Low-Rank (LoRA)

O fine-tuning consiste en continuar o adestramento dun modelo xa pre-adestrado cun pequeno conxunto de datos específico para especializalo nunha tarefa concreta. Nos LLM isto implica alimentar o modelo con exemplos do dominio (p.ex. diálogos de soporte técnico) e buscar unha pequena modificación dos pesos que minimice a perda partindo dos pesos orixinais. Actualizalo todo é asumible en modelos medianos, pero inviable nos actuais xigantes,

porque duplicaría o almacenamento e o cómputo. Por iso xurdiron técnicas de adaptación eficiente, que axustan só unha fracción dos parámetros e fan o proceso moito máis lixeiro.

Adaptación Low-Rank (LoRA): LoRA é unha destas técnicas eficientes de fine-tuning onde a idea central é conxelar os pesos orixinais do modelo pre-adestrado e aprender únicamente pequenas matrices auxiliares de baixo rango que se engadirán a eses pesos durante a adaptación. Isto permite axustar só unha fracción mínima dos parámetros, reducindo o consumo de memoria e cómputo en varios ordés de magnitude, manter modelos especializados moi lixeiros e, ademais, fusionar esas pequenas actualizacións cos pesos orixinais para conservar a mesma velocidade de inferencia sen modificar a arquitectura.

LoRA é unha técnica que consegue igualar ou incluso mellorar o rendemento do fine-tuning convencional en múltiples taredas de NLP e presenta diferentes vantaxes como son:

- **Eficiencia en almacenamento e cómputo:**Só se gardan unhas cantas matrices pequenas por tarefa, polo que é factible manter múltiples versións fin-tuneadas do modelo sen duplicar todo o LLM. Así, distintos LoRA modules pódense intercambiar sobre un mesmo modelo base, habilitando a conmutación rápida de tarefas con mínimos requisitos de memoria.

- **Menor necesidade de hardware de alto custo:** Ó reducir drásticamente os parámetros que se optimizan, diminúe tamén o uso de memoria e a carga computacional por batch, permitindo adestrar modelos de gran tamaño con GPUs máis modestas ou en menos tempo. Isto democratiza un pouco o fine-tuning de LLMs, antes restrinxido a grandes corporacións con infraestruturas masivas.

- **Ausencia de latencia extra en inferencia:** A solución de LoRA é algebraicamente simple e pódese integrar sumando as matrices aprendidas ós pesos, polo que, unha vez despregado, o modelo funciona igual de rápido cá versión orixinal, a diferenza doutras técnicas como adapters que engaden capas e retardos

Pola contra, LoRA tamén ten algunhas limitacións. A principal é que require definir de antemán en que capas/matrizes se aplicará a adaptación e elixir o rango r adecuado; un valor moi baixo de r pode ser insuficiente para capturar a adaptación requirida (underfitting)

Aplicación no proxecto (fine-tuning de LLaMA con LoRA):

Neste TFM utilizouse LoRA para afinar un modelo LLaMA pre-adestrado co obxectivo de adaptalo ó dominio das auditorías de ciberseguridade. En concreto, partimos de LLaMA-3.1-8B, un LLM xeral de código aberto, e realizamos un fine-tuning empregando un conxunto de datos especializado: preguntas frecuentes sobre normativas de seguridade, descricións de controis de auditoría, informes técnicos e outras fontes relevantes recopiladas. O resultado do fine-tuning é un modelo LLaMA especializado que conserva o coñecemento e fluidez do LLM

xeral, pero que está moito mellor sintonizado co vocabulario e os contidos de ciberseguridade. Importante destacar que, coa técnica de LoRA, podemos distribuír o chatbot open-source sen incluír directamente os pesos completos de LLaMA. Isto simplifica cuestións de licenza e distribución, á vez que mantén a calidade do sistema. En resumo, LoRA permitiu adaptar eficazmente o corazón do chatbot ao dominio de ciberseguridade, superando as limitacións de recursos e garantindo un tempo de inferencia practicamente idéntico ao modelo base, é dicir, sen penalización na experiencia do usuario final.

2.4 Pipeline de Retrieval-Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) é unha técnica que combina modelos de linguaxe xerativos con motores de recuperación de información para producir respostas máis precisas, actualizadas e fundamentadas en coñecemento externo. aínda que estes modelos conteñen unha enorme memoria paramétrica (o aprendido durante o adestramento), poden carecer de coñecemento detallado ou recente sobre certos temas. RAG aborda este problema proporcionando ao LLM información puntual extraída dunha memoria non-paramétrica, é dicir, dun conxunto de datos ou documentos externos. En palabras sinxelas, en lugar de pedirlle ao modelo que responda unicamente co que "sabe" polos seus parámetros (que podería estar incompleto ou desactualizado), permítese que busque nunha base de coñecemento e extraia de alí feitos concretos para apoiar a súa xeración. En esencia, un sistema RAG consta de dúas compoñentes:

- **un módulo de retrieval (recuperación)** que, dada a pregunta do usuario, atopa os documentos ou fragmentos de texto máis relevantes nun corpus
- **un módulo de generation (xeración)** normalmente un LLM que elabora a resposta final combinando a pregunta e o contido recuperado.

Arquitectura e funcionamento interno: Unha pipeline RAG típica funciona en dúas fases. Primeiro, na fase de indexación (offline), prepárase a base de coñecemento: recóllense os documentos fontes e procésanse en fragmentos manexables. A continuación, cada fragmento é codificado nunha representación matemática eficiente, habitualmente un vector de embedding de dimensión fixa que captura o seu contido semántico. Estes vectores almacénanse nun índice especial (unha base de datos vectorial ou un índice invertido combinado con vectores) xunto cun identificador do fragmento orixinal. Segundo, na fase de pregunta-resposta (online), cando chega unha consulta do usuario, o sistema segue estes pasos secuencialmente:

- **Formulación da consulta:** Tómase o texto da pregunta do usuario e obtense tamén o seu embedding vectorial mediante o mesmo modelo que se usou para os documentos (isto garante que estean no mesmo espazo semántico).

- **Recuperación de documentos:**Búscanse no índice aqueles fragmentos cuxos vectores sexan máis semellantes ao vector da consulta, normalmente empregando similitude do coseno ou distancia euclidiana para tal comparación. En termos prácticos, recupéranse os top K documentos máis relevantes que teoricamente conteñen información relacionada coa consulta. Esta procura pode complementarse con filtros ou métodos lexicográficos.
- **Combinación consulta-contexto:** Os fragmentos recuperados extráense da base de datos e combínanse coa pregunta do usuario para formar a entrada extendida que se lle proporcionará ao modelo xerativo.
- **Xeración da resposta:** Finalmente, pásase este prompt aumentado ao LLM. O LLM procesa tanto a pregunta coma o contexto achegado e xera unha resposta en linguaxe natural que idealmente fai uso da información dos documentos para dar unha contestación específica, correcta e con referencias. Se está ben deseñado o prompt, o LLM pode incluso incluír fragmentos citados dos documentos ou referencias bibliográficas na súa resposta.

Este procedemento asegura que o LLM ten acceso a unha memoria externa actualizada en lugar de confiar unicamente na súa memoria de adestramento.

Vantaxes de RAG: Integrar recuperación máis xeración ofrece varios beneficios clave fronte ao uso dun LLM illado.

- **Mellorar a precisión factual e minimiza alucinacións** xa que o modelo basea a súa resposta en documentos concretos. Ao dispoñer de datos reais durante a inferencia, redúcese a probabilidade de que o LLM "invente" respostas; de feito, proporciónaselle evidencia que pode citar, aumentando a confiabilidad da resposta.
- **Capacidade de incorporación coñecemento novo ou personalizado** sen re-adestrar o LLM. Mentres que actualizar un modelo enorme cos últimos datos pode ser inviable, en RAG abonda con engadir os novos documentos ao índice. Dito doutra forma, a pipeline pode ampliar dinámicamente o mundo do LLM engadindo ou modificando a base documental, facendo que o sistema responda con información en tempo real. Isto resulta crítico nun campo como o da ciberseguridade, onde constantemente xorden ameazas e actualizacións: a RAG fai posible que o chatbot incorpore esas fontes novas durante a consulta.
- **Contribúe á transparencia e trazabilidade das respostas** como o LLM fundamenta as súas saídas en documentos recuperados, pódense devolver esas referencias ao usuario de forma similar a como un informe humano incluiría as fontes. Isto xera máis confianza no usuario, que pode verificar a orixe dos datos.

- **Eficiencia e flexibilidade** permitindo reutilizar un mesmo LLM xeral para múltiples dominios simplemente cambiando o conxunto de documentos. Comparado con entrenar modelos especializados, RAG adoita ser moito máis rápido e barato de implementar facilitando probar diferentes fontes de datos sen modificar a arquitectura central.

Porén, tamén podemos localizar varias limitacións como que a calidade da busqueda pode comprometerse se non ten os datos correctos e aumentar o risco de alucinacións, o problema de calcular os tamaños correctos dos fragmentos de contextos, a coherencia narrativa da mezcla de diferentes fontes para integrar a información e o rendemento, xa que este proceso engade latencia que pode reducirse con índices en memoria, cachés e optimizacións.

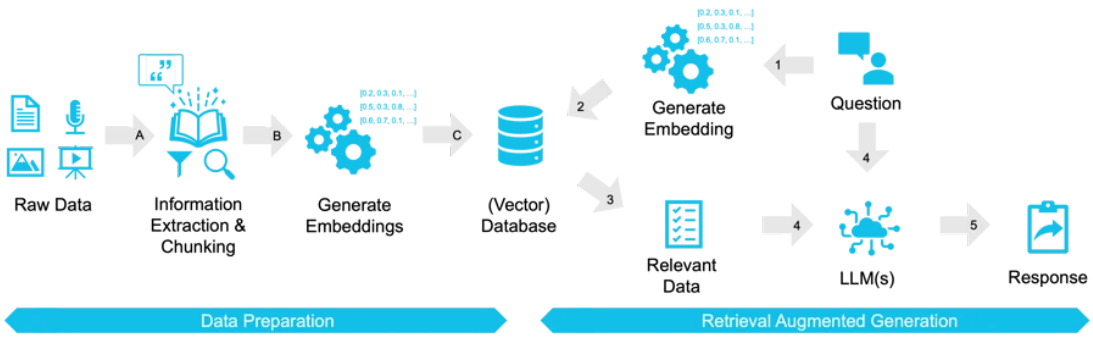

Figura 2.3: Esquema do funcionamiento da pipeline RAG.

Implementación no noso proxecto (LangChain + ElasticSearch):

O chatbot de auditoría de ciberseguridade desenvolvido integra unha pipeline RAG usando a libraría LangChain e o motor ElasticSearch como infraestrutura base. LangChain proporciónanos unha interface de alto nivel para encadear os pasos descritos: define cadeas onde a saída dunha acción alimenta a seguinte. Pola súa banda, ElasticSearch actúa como base de coñecemento documental. Optouse por ElasticSearch pola súa flexibilidade e desempeño na recuperación: este motor soporta tanto busca tradicional por texto (co seu potente índice invertido) como capacidades de vector search recentes para embeddings, combinando ambos enfoques se se desexa. A aplicación emprega un índice en ElasticSearch que garda unha colección ampla de información do ámbito da ciberseguridade, previamente fragmentados e convertidos en vectores mediante un modelo de embeddings. ElasticSearch almacena estes vectores xunto cos textos e permite recuperalos por semellanza; cando o usuario lanza unha pregunta, LangChain calcula o seu embedding, busca os fragmentos máis próximos e insire ese contexto no prompt enviado ao LLM, de maneira que o modelo pode responder con precisión e apoiar as súas explicacións na información recuperada en lugar de depender unicamente do seu coñecemento interno.

2.5 Representacións vectoriais (Embeddings)

Definición e motivación

As representacións vectoriais —coñecidas habitualmente como embeddings— son códigos numéricos de dimensión fixa que permiten describir entidades discretas (palabras, frases, documentos, imaxes, usuarios, etc.) nun espazo continuo. A idea central é que a proximidade xeométrica entre dous vectores se corresponda coa semellanza conceptual das entidades ás que representan. Deste xeito, un algoritmo pode operar con semántica mediante álxebra lineal, o que facilita tarefas de busca, clasificación ou xeración de contido.

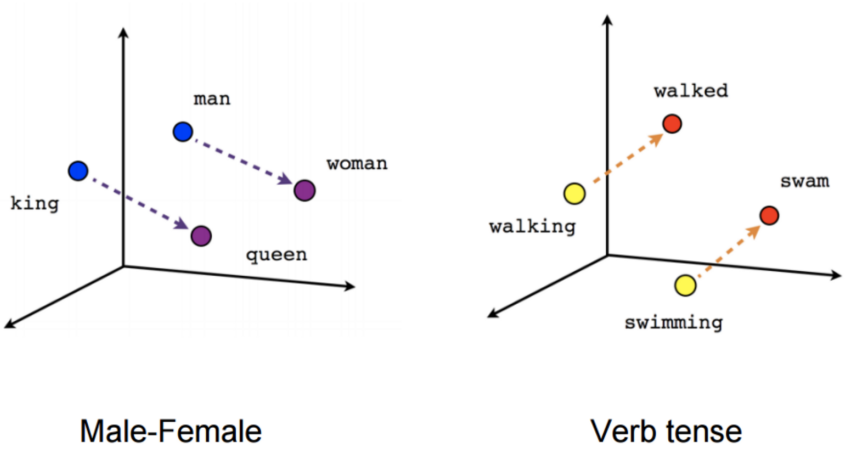

Figura 2.4: Exemplo simple de embeddings nun eixo.

O punto de partida teórico é a hipótese distribucional da lingüística ("os termos que ocorren en contextos semellantes teñen significados semellantes"). Os primeiros enfoques —como o Latent Semantic Analysis dos anos 90— aproximaban esta idea reducindo grandes matrices termo-documento mediante técnicas de álxebra (p. ex. SVD). A revolución chegou cando se introduciu o adestramento de redes neuronais especificamente orientadas a aprender estes vectores, popularizado por Word2Vec. Desde entón sucedéronse métodos análogos (GloVe, FastText, etc.) e, máis recentemente, modelos de tipo transformer (BERT, RoBERTa, Llama, ...) que producen representacións aínda máis ricas.

Os métodos seguiros a seguinte evolución:

- **Embeddings estáticos de palabra** Cada termo do vocabulario recibe un único vector denso, aprendido a partir dun gran corpus. Estes vectores capturan relacións léxicas simples ("rei" ↔ "raíña") e permiten operacións aritméticas que reflicten patróns de xénero, tempo verbal, localización, etc.
- **Embeddings contextuais** Coas arquitecturas transformer, unha mesma palabra ad-

quire distintas representacións segundo o contexto (“banco” como moble vs. “banco” como entidade financeira). Isto resolve ambigüidades sen recorrer a regras ad-hoc e mellora o desempeño en tarefas de comprensión.

- **Embeddings de nivel frase/documento** Pasar da palabra á sentenza non se resolve simplemente sumando vectores. Modelos específicos —Sentence-BERT, MPNet, MiniLM, entre outros— axústanse para xerar un único vector representativo dun texto completo. Estes sentence embeddings fan posible medir semellanza entre preguntas e respostas, agrupar parágrafos semellantes ou recuperar contido relevante nunha arquitectura RAG.

Unha vez obtidos os vectores compre utilizar unha función para estimar a proximidade entre eles e pode ser mediante:

- **Similitude de coseno:** mide o coseno do ángulo entre dous vectores, polo que depende só da dirección e non da magnitude, os valores próximos a 1 implican contido moi parecido.
- **Distancia euclídea:** medida entre dous puntos dun espazo cartesiano n-dimensional, calculada como a raíz cadrada da suma dos cadrados das diferenzas en cada coordenada. Intuitivamente, corresponde á lonxitude real do segmento que une ambos puntos, tal e como medirías cunha regra no plano ou cunha cinta métrica no espazo.

Con todo esto podemos utilizar os embeddings para aproveitar as vantaxes principais que serían:

- **Captura de semántica e xeneralización:** permitindo que palabras ou frases con significado relacionado teñan representacións próximas, aínda que non compartan literalmente caracteres.
- **Redución de dimensionalidade:** En lugar de traballar nun espazo inmenso igual ao tamaño do vocabulario (onde cada palabra é independente), os embeddings traballan nun espazo de dimensión moderada (centos de dimensións) onde as relacións están codificadas

aínda que poden presentar outras como a capacidade de transferir información ou a operatividade coa álxebra lineal.

Embeddings no noso proxecto:

No desenvolvemento do chatbot de auditoría, as representacións vectoriais son unha peza fundamental para habilitar a recuperación de información (retrieval) por semellanza de significado. Así, cando o usuario formule unha pregunta nunha redacción libre – que pode non

coincidir exactamente cos termos nos documentos – o sistema seguirá sendo quen de atopar os textos relevantes. Isto supón unha gran mellora fronte a unha procura booleana tradicional: en lugar de depender só de palabras clave, estamos efectivamente buscando por conceptos. Na práctica, comprobouse que o uso de embeddings permitiu atopar información aínda que a formulación variase. Outro factora ter en conta é que as representacións vectoriais permiten que os resultados veñan ordenados por grado de similitude, axudando a xestionar casos onde non hai unha correspondencia perfecta pero hai varios documentos parcialmente relacionados.

2.6 Open Source

Modelo de desenvolvemento de software no que o código fonte está publicamente dispoñible baixo unha licenza que garante tres liberdades básicas: examinar, modificar e redistribuír. Este paradigma fomenta a transparencia e a seguridade, pois calquera pode auditar o código e detectar vulnerabilidades. A colaboración global posibilita unha innovación máis rápida: milleiros de persoas, comunidades e empresas contribúen correccións e novas funcionalidades, repartindo custos de desenvolvemento. Ademais, promove a soberanía tecnolóxica: quen adopta software aberto evita depender dun único provedor e pode adaptalo ás súas necesidades sen restricións. Entre os retos destacan a sostibilidade dos proxectos, moitos dependen de voluntariado, e a correcta xestión de licenzas en contornas corporativas para evitar incumprimentos. Con todo, open source consolidouse como un motor clave da economía dixital: desde o núcleo Linux ata linguaxes como Python ou frameworks como Django, gran parte da infraestrutura de Internet e da intelixencia artificial actual baséase en software aberto, demostrando que compartir coñecemento acelera o progreso tecnolóxico.

Capítulo 3

Tecnoloxías e ferramentas empregadas

3.1 Linguaxes de programación

O desenvolvemento do proxecto guardIAn baséase na integración de distintos compoñentes software que requiren linguaxes de programación específicos segundo a súa función dentro da arquitectura da aplicación. Principalmente empregamos Python e JavaScript (con JSX), seleccionadas pola súa ampla comunidade, compatibilidade coas ferramentas utilizadas e versatilidade para construír tanto sistemas baseados en intelixencia artificial como interfaces web modernas e funcionais.

3.1.1 Python

Python foi o linguaxe de programación principal no backend do proxecto, desempeñando un papel esencial en varias capas da aplicación:

- **Desenvolvemento da API REST:** Utilizouse o framework FastAPI, que permite crear servizos web asincrónicos, rápidos e facilmente documentables. Esta API é o nexo de unión entre a interface web e o modelo de linguaxe, xestionando as peticións e respostas de maneira eficiente.
- **Xestión e adaptación do modelo de linguaxe:** Python tamén foi empregado para o proceso de fine-tuning do modelo LLaMA 3.1 utilizando técnicas como LoRA (Low-Rank Adaptation), así como para a integración coa pipeline RAG (Retrieval-Augmented Generation) mediante a libraría LangChain. Estas tarefas fan uso de librarías específicas como Transformers, HuggingFace, bitsandbytes ou Accelerate.
- **Indexación e busca de documentos:** A creación de índices densos a partir dos datos

extraídos de foros de ciberseguridade foi realizada mediante Python, conectando cos servizos de ElasticSearch para almacenar e recuperar información relevante durante as consultas ao chatbot.

- **Extracción e procesamento de datos:** Un aspecto fundamental do proxecto foi a construción dun dataset de calidade a partir de fontes públicas. Para iso empregouse o framework Scrapy, que permitiu desenvolver spiders específicos para foros como Gentoo e Debian co fin de ter varias fontes de datos. Estes spiders foron deseñados para navegar automaticamente polas páxinas, identificar dúbidas resoltas no foro relacionadas ca ciberseguridade e extraer información textual estruturada. Unha vez recollidos os datos, procesáronse mediante scripts auxiliares en Python para realizar tarefas como limpeza, normalización e filtrado, garantindo a súa relevancia e consistencia antes de utilizalos no adestramento e indexación.

- **Scripts auxiliares e tarefas automatizadas:** Python tamén foi usado para outras tarefas secundarias como a obtención de información do conxunto de datos e procesado a fondo dos mesmos.

A elección de Python responde á súa excelente integración coas tecnoloxías de intelixencia artificial (aínda que outras linguaxes como Java cada vez cobran máis forza seguen estando moi lonxe de ter toda a contorna de librerías de Python), a dispoñibilidade de librarías de alto nivel e o soporte comunitario, que o converten nun estándar de facto no ámbito da IA aplicada a día de hoxe.

3.1.2 JavaScript (JSX con React)

No lado do frontend, optouse por React como framework principal, empregando JavaScript con sintaxe JSX para o desenvolvemento dunha interface web moderna, interactiva e modular. React permite construír compoñentes reutilizables que xestionan de forma eficiente o estado da aplicación e a interacción co usuario. Isto foi especialmente útil para:

- Crear formulares de entrada de texto e uploads de arquivos (logs ou código a analizar) que interactúan directamente coa API.

- Mostrar respostas xeradas polo chatbot, incluíndo mensaxes explicativas, análises de rexistros ou recomendacións técnicas.

- Xestionar o estado global da sesión, gardando o contexto da conversación para permitir un diálogo continuo e coherente co asistente mediante o envío ao backend do contexto no prompt.

A estrutura en JSX simplifica a mestura de lóxica JavaScript e estrutura HTML, permitindo un desenvolvemento máis claro e intuitivo da interface además de poder utilizar ferramentas como React Router, hooks e contextos que facilitaron a navegación entre páxinas e a xestión eficiente de estados e efectos secundarios (por exemplo, peticións asincrónicas á API con fetch ou axios). A súa combinación con Python no backend supón unha arquitectura desacoplada pero perfectamente interoperable, baseada en estándares abertos, permitindo unha alta escalabilidade e mantemento da aplicación.

3.2 Frameworks e librarías

O desenvolvemento de guardIAn requireu o uso combinado de varios frameworks e librarías que permiten implementar tanto a capa de intelixencia artificial como o backend e o frontend da aplicación. Estas ferramentas foron escollidas pola súa eficiencia, integración cos estándares actuais e ampla aceptación na comunidade de desenvolvedores. A continuación descríbense os principais compoñentes utilizados.

3.2.1 LangChain

LangChain é un framework especializado no desenvolvemento de aplicacións baseadas en modelos de linguaxe, especialmente útil para sistemas que combinan xeración de texto con recuperación de información (RAG, Retrieval-Augmented Generation). No contexto de guardIAn, empregouse para integrar o modelo LLaMA 3.1 con fontes externas de coñecemento mediante índices vectoriais, así como para construír cadeas de razonamento que permiten combinar consultas aos datos con respostas xeradas de forma natural. LangChain facilitou a abstracción do funcionamento interno do modelo e a estruturación do fluxo de datos entre as distintas capas do sistema, como o modelo, o índice e a API. A súa modularidade resultou clave para deseñar unha arquitectura flexible e extensible para o chatbot, permitindo a incorporación sinxela de novas fontes de datos ou funcionalidades no futuro.

3.2.2 FastAPI

FastAPI é un framework moderno e asincrónico para a creación de APIs RESTful en Python, empregado como base para o backend da aplicación. A súa arquitectura permite definir puntos de entrada HTTP de maneira robusta e clara, ao tempo que facilita a validación automática dos datos de entrada mediante o uso de anotacións de tipo. Destaca pola súa integración sinxela con ferramentas de testeo como Postman e pola súa alta eficiencia no manexo de múltiples peticións simultáneas, grazas ao seu deseño baseado en async/await. A API construída con FastAPI serve de ponte entre a interface web desenvolvida en React e o modelo de linguaxe, asegurando unha comunicación fluída e segura entre ambos compoñentes.

3.2.3 Django

Django é un framework web de alto nivel desenvolvido en Python que se empregou neste proxecto para facilitar o desenvolvemento rápido dunha aplicación web segura, escalable e ben estruturada. Utilizouse como base para xestionar o backend da web, servindo de estrutura fundamental do sitio, así como para implementar funcionalidades administrativas e de autenticación que poidan ser necesarias nunha futura expansión. Aproveitouse o seu robusto sistema de modelos, que permite unha integración eficiente con bases de datos relacionais e facilita a posible ampliación da aplicación. En conxunto, Django complementa a arquitectura global do sistema, ofrecendo unha infraestrutura sólida e profesional sobre a que construír.

3.2.4 React

React é unha libraría de código aberto para JavaScript que permite construír interfaces de usuario dinámicas, modernas e responsive de forma declarativa e baseada en compoñentes reutilizables. O seu núcleo se apoia nun fluxo de datos unidireccional e nun virtual DOM que optimiza as actualizacións da pantalla, ofrecendo alta eficiencia e rendemento mesmo en aplicacións interactivas. Dispón dun ecosistema amplo con funcionalidades como Hooks para xestión de estado e efectos, React Router para navegación en aplicacións de páxina única e compatibilidade con TypeScript para tipado estático. A súa popularidade e forte comunidade fan que conte con abundantes recursos, extensións e boas prácticas, converténdoa nunha opción robusta e escalábel para o desenvolvemento de frontends modernos.

3.2.5 Transformers

A libraría Transformers, desenvolvida por HuggingFace, constitúe a ferramenta principal empregada para traballar con modelos de linguaxe como LLaMA neste proxecto. A súa versatilidade permitiu cargar modelos preentrenados e adaptalos ao dominio específico da ciberseguridade, así como xestionar o proceso de inferencia e a integración co resto da pipeline. Facilitou o traballo con outras librarías complementarias como Accelerate, bitsandbytes e PEFT, co obxectivo de optimizar o rendemento. Grazas á súa ampla compatibilidade con diversos modelos e á capacidade para xestionar con eficiencia pipelines de procesamento de linguaxe natural, Transformers converteuse nun compoñente esencial para o desenvolvemento do proxecto.

3.2.6 Scrapy

Figura 3.1: Logotipo de Scrapy e Python.

Scrapy é un framework para o desenvolvemento de web crawlers e scrapers en Python que foi empregado neste traballo para crear spiders personalizados capaces de extraer contido relevante de foros especializados en ciberseguridade, como o foro de Gentoo. A ferramenta permitiu a navegación automatizada por páxinas estruturadas, identificando publicacións, respostas e contido técnico de interese. Ademais, facilitou a exportación dos datos recollidos en formatos manexables como JSON, o que permitiu o seu posterior procesamento e uso tanto na creación dos índices como no fine-tuning do modelo. Revelou ser unha ferramenta fundamental para alimentar a base de coñecemento do chatbot con datos reais e actualizados.

3.3 Ferramentas de desenvolvemento

3.3.1 Taiga

Taiga é unha plataforma de xestión de proxectos áxiles de código aberto concibida para facilitar a vida a equipos pequenos e medianos que traballan con metodoloxías como Scrum ou Kanban, cunha interface visual limpa e intuitiva, a súa flexibilidade permite organizar o backlog, priorizar historias de usuario, planificar sprints, xestionar "issues" e medir a velocidade nun único contorno colaborativo. Nun único contorno web permite crear e priorizar historias de usuario, estimar a súa complexidade mediante puntos de historia e asignalas aos distintos sprints ou columnas do taboleiro Kanban, de maneira que se poida ter unha fotografía instantánea do estado do traballo. Taiga está distribuída baixo licenza Mozilla Public License 2.0, o que garante que o seu código fonte é totalmente aberto, auditable e extensible, de xeito que se pode despregar na nube propia da organización ou customizar a vontade sen depender dun provedor pechado seguindo moi ben a filosofía open-source deste traballo.

3.3.2 GanttProject

GanttProject é unha aplicación de escritorio de código aberto orientada á planificación e seguemento de proxectos mediante diagramas de Gantt, a clásica representación temporal

na que cada tarefa ocupa unha barra proporcional á súa duración e dependencias o que a fai moi intuitiva e visual. A ferramenta permite definir o calendario global (xornadas laborais, días festivos), crear hitos, tarefas jerárquicas e relacións de precedencia (fin-a-inicio, inicio-a-inicio, etc.), asignar recursos humanos ou materiais con porcentaxes de carga, estimar custos, e visualizar automáticamente a ruta crítica; todo iso facilita detectar pescozos de botella e reaxustar prazos con só arrastrar barras no cronograma. Está distribuído baixo licenza GPL, GanttProject non impón restricións comerciais e pode ser adaptado ou integrado libremente con outras ferramentas, resultando especialmente útil en contornos que non requiren toda a complexidade dun ERP profesional pero precisan controlar prazos, custos e dependencias de maneira visual e sinxela.

3.3.3 GitHub

Plataforma web de hospedaxe de repositorios Git que combina control de versións distribuído cun amplo conxunto de servizos colaborativos orientados a desenvolvedores e equipos de software. Nun único espazo permite almacenar código fonte, xestionar ramas e pull requests para revisar cambios, rexistrar issues ou tarefas, documentar proxectos con wikis, e crear fluxos de integración e despregamento continuo mediante GitHub Actions, todo iso con historial completo e trazabilidade de cada contribución. Grazas ao seu sistema de permisos granular e á opción de repositorios públicos ou privados, GitHub é apto tanto para proxectos de código aberto—onde a "social coding" facilita atopar colaboradores, clonar, forkar e enviar melloras—como para aplicacións empresariais que requiren confidencialidade, auditoría de cambios e políticas de seguridade avanzadas. A interfaz web, simplifica tarefas que en Git resultarían máis laboriosas dende a liña de comandos, mentres que a integración nativa con ferramentas externas crea un ecosistema no que se poden automatizar probas, compilar contedores, versionar documentación ou publicar paquetes. Esta plataforma converteuse nun estándar de facto para millóns de desenvolvedores ao facilitar a colaboración global, a transparencia e a fiabilidade nos ciclos de vida do software.

3.3.4 Postman

Postman é unha aplicación que serve, sobre todo, para probar se unha API funciona correctamente. Permíteche escribir a URL dun endpoint, escoller o método (GET, POST, PUT, DELETE…), engadir cabeceiras ou corpo en JSON, premer "Send" e ver ao instante a resposta que devolve o servidor: código de estado (200, 404…), tempo de execución, cabeceiras e contido. Así comprobamos se os parámetros son os correctos, se a autenticación é válida ou se os datos chegan ben antes de incluír chamadas á API do código. Ademais, podese gardar coleccións de peticións, engadir test automáticos (por exemplo, verificar que a resposta leva un 200 e un campo concreto en JSON) e compartir eses casos con outras persoas para que todos

comproben o mesmo. En resumo, Postman actúa como un "laboratorio" onde experimentar e depurar chamadas á API dun xeito rápido, visual e sen necesidade de programar scripts separados.

3.3.5 Visual Studio Code

Visual Studio Code é un editor de código fonte lixeiro, gratuíto e multiplataforma (Windows, macOS e Linux) que combina unha interface minimalista con potentes funcións: resaltado de sintaxe para centos de linguaxes, completado intelixente, depuración integrada, terminal embebido, xestión de control de versións Git e un enorme mercado de extensións que permiten engadir soporte para frameworks, contedores ou diversas ferramentas, todo sen sair do editor.

Capítulo 4

Planificación e metodoloxía

Neste capítulo detállase a metodoloxía e a xestión empregadas para o desenvolvemento xustificando a elección dunha metodoloxía áxil do tipo Scrum fronte a enfoques en cascada máis ríxidos. Explicase como, dado o carácter e a complexidade técnica do proxecto, resultaba esencial dispor dun método iterativo e flexible que permitise entregar funcionalidades incrementais con frecuencia, recibir feedback continuo e adaptar o alcance ás dificultades que xorden ao facer o traballo. Ademais, descríbese con detalle a adaptación de Scrum a un contexto dun proxecto individual, onde os directores asumen o papel de Product Owner e o alumno asume os roles de Scrum Master e desenvolvedor, poñendo de relevo como se organizaron os sprints de dúas semanas, a planificación, o seguimento e as retrospectivas para manter un progreso sostido. Neste sentido, recóllense as estimacións temporais e en puntos de historia para cada sprint, amosando como se valoraron as tarefas principais e como se axustou a velocidade de traballo ao longo do proxecto. Tamén se describen os recursos humanos, materiais e de software que permitiron construír a solución de xeito eficaz e cun orzamento reducido grazas ao uso de tecnoloxías open-source e unha estimación de custos e cunha análise de riscos salientando as principais ameazas, xunto cos plans de mitigación e continxencia implementados para garantir o éxito do proxecto dentro do calendario académico.

4.1 Metodoloxía de traballo

Unha parte fundamental do traballo foi a escolla da metodoloxía utilizada, xa que é a que pode garantir o éxito ou o fracaso do mesmo axudándonos a ter en conta diversos factores como os riscos, unha visión estratéxica e toda a xestión completa, valoráronse diferentes metodoloxías pero ao final decidiuse utilizar Scum que é unha metodoloxía áxil incremental con ciclos de vida iterativos resultando moi flexibel. Os factores principais que se tiveron en conta para esta escolla foron:

- **Iteración e adaptación continua:** Esta metodoloxía permite avanzar en ciclos cur-

tos que favorecen a aprendizaxe constante e axustan o rumbo do proxecto conforme se obteñen novos coñecementos, facilitando tamén a detección temperá de posibles desviacións ou puntos débiles a través de entregas parciais que van mellorando progresivamente o produto.

- **Priorización e xestión de tarefas complexas:** organízase dividindo o traballo en pequenos bloques (historias de usuario) aos que se achega unha estimación de esforzo, o que permite planificar de xeito realista canto tempo require cada función e evitar sorpresas. Ao mesmo tempo, a revisión sistemática das tarefas antes de cada iteración garante que sempre se adiquen recursos primeiro ás actividades de maior impacto, evitando así perder tempo en funcionalidades que non sexan imprescindibles nas fases iniciais. Deste xeito, mantense unha visión clara de que debe facerse a continuación e como guiar o avance do proxecto de xeito eficiente e ordenado.
- **Mitigación de riscos técnicos e de planificación:** Adoptar ciclos curtos de traballo permite identificar rapidamente calquera imprevisto ou desvío no proxecto, o que facilita axustar o plan sen comprometer o conxunto do cronograma; ao mesmo tempo, esta planificación por etapas axuda a distribuír e coordinar mellor os recursos dispoñibles, evitando sobrecargas nalgúns momentos e garantindo que cada tarefa conte coas capacidades necesarias para o seu desenvolvemento.
- **Rol único do estudante e colaboración con tutores:** o estudante asume a totalidade das funcións organizativas e de desenvolvemento do proxecto, adaptando a metodoloxía ás súas necesidades individuais, mentres que os tutores actúan como guías estratéxicos que establecen as liñas mestras e revisan periódicamente os avances para orientarme nas prioridades e cambios de alcance.
- **Visión modular e incremental do produto final:** O proxecto concíbese como un conxunto de módulos interdependentes que se van construíndo e comprobando de forma progressiva, de modo que cada parte funcional permita obter resultados parciais que poidan ser validados polos expertos no ámbito da ciberseguridade antes de avanzar ao seguinte bloque. Deste xeito, garante que cada compoñente sexa probada de maneira autónoma e se poidan incorporar axustes continuos, asegurando que o produto final cumpra as necesidades reais de auditoría de forma áxil e efectiva.

4.1.1 Scrum

Como se mencionou anteriormente, Scrum é unha metodoloxía áxil pensado para desenvolver produtos complexos de maneira iterativa e incremental. Basada en ciclos curtos de tempo chamados Sprints, nos que o equipo entrega unha versión funcional (incremento) do

produto, obtén retroalimentación e axusta o plan antes de comezar o seguinte ciclo. A transparencia, a inspección continua e a adaptación rápida constitúen os seus piares fundamentais mais aínda así segue a ser máis complexa, tendo diferentes atributos como son os artefactos, o equipo e os eventos que xogan un papel fundamental.

4.1.2 Artefactos

Oss artefactos son elementos que proporcionan visibilidade do traballo e fomentan a transparencia durante o desenvolvemento. Cada artefacto reflicte un aspecto clave do proceso, permitindo ao equipo e aos stakeholders comprender o estado actual do produto e do Sprint.

- **Product Backlog:** Rexistra e prioriza os requisitos desde o punto de vista do cliente. Empeza cunha visión inicial do produto e crece e evoluciona ao longo do desenvolvemento. Os elementos do Product Backlog adoitan denominarse "historias de usuario" e configuranse segundo o valor de negocio, asegurando que o equipo sempre traballa no máis importante primeiro.

- **Sprint Backlog:** Rexistro dos requisitos desde o punto de vista dos desenvolvedores para un Sprint concreto. É a lista de tarefas (divididas a partir das historias de usuario elixidas do Product Backlog) que se deben completar durante o Sprint para acadar o incremento previsto. Inclúe tanto as tarefas en curso como as estimacións de esforzo e o obxectivo do Sprint.

- **Incremento:** Resultado de cada Sprint. Conxunto de todos os elementos do Product Backlog completados durante o Sprint que cumpren a Definition of Done. O Incremento supón valor engadido ao produto e pode ser entregable ou demostrábel ao final do Sprint, garantindo que cada iteración achega unha versión funcional máis próxima ao produto final.

4.1.3 Equipo

Conxunto autoorganizado e multifuncional de persoas responsable de entregar incrementos de produto de forma iterativa. Normalmente está composto por 3 a 9 membros coas habilidades necesarias para completar o traballo sen dependencia externa.

- **Product Owner:** Garante que o equipo Scrum traballe de xeito axeitado dende a perspectiva do negocio. Axuda ao usuario a redactar as historias de usuario, establece a súa prioridade e garda o Product Backlog sempre actualizado. É o único responsable de maximizar o valor do produto e de aclarar requisitos ao equipo.

- **Scrum Master:** Actúa como facilitador e gardián das regras do marco Scrum. Aségurase de que toda a organización entenda e respecte estas normas, e elimina os obstáculos que impiden o progreso do Sprint. Proporciona asesoramento e formación tanto ao Product Owner como ao equipo de desenvolvedores para mellorar continuamente procesos e prácticas.
- **Desenvolvedor:** Calquera membro do equipo que contribúe directamente á entrega do incremento en cada Sprint. Traballan de forma colaborativa reutilizando habilidades transversais (análise, deseño, codificación, probas, documentación). O seu obxectivo é completar as historias de usuario comprometidas e garantir a calidade do Incremento.

4.1.4 Eventos

Existen diferentes tipos de eventos en Scrum que son reunións estruturadas e periódicas que permiten ao equipo inspeccionar o traballo realizado, adaptar o plan e mellorar continuamente tanto o produto como o proceso. Cada evento ten unha finalidade concreta para asegurar a transparencia, a colaboración e a entrega de valor.

- **Sprint Review:** Realízase ao final do Sprint para inspeccionar o Incremento desenvolvido e adaptarse ao mercado ou aos feedbacks do cliente. O equipo amosa o que rematou, recolle comentarios de stakeholders (clientes, usuarios, Product Owner) e actualiza o Product Backlog coa información recollida para enriquecer futuros Sprints.
- **Sprint Planning:** Reunión ao inicio de cada Sprint na que o equipo revisa o Product Backlog para seleccionar as historias de usuario que se comprometerán a completar no Sprint. Define o obxectivo do Sprint e estiman as tarefas necesarias, asegurando que teñan unha visión compartida do que van entregar.
- **Sprint Retrospective:** Celebración interna do Scrum Team despois da Sprint Review, enfocada en inspeccionar o proceso e mellorar continuamente. Analízanse aspectos que foron ben e identifican oportunidades de mellora (técnicas, comunicación, ferramentas). O resultado é un plan de acción para implementar cambios no próximo Sprint.
- **Daily Scrum:** Reunión diaria, de ata 15 minutos, na que o equipo de desenvolvemento sincroniza actividades e planifica as próximas 24 horas. Cada membro responde brevemente a tres preguntas: que fixo onte, que fará hoxe e que impedimentos ten. Serve para identificar bloqueos e axustar pequenos cambios no día a día.

4.1.5 Elementos de control e calidade

Conxunto de prácticas, criterios e ferramentas empregados para supervisar o progreso do desenvolvemento e garantizar que o produto final cumpra os estándares de calidade acorda-

dos. Inclúen tanto mecanismos de inspección continua, como gráficos ou reunións de revisión, como definicións explícitas de aceptación que aseguran que cada iteración entrega valor fiable e cumprible.

- **Burn down chart:** Gráfica diaria que reflexa de forma pública a evolución do traballo pendente no Sprint. Cada punto amosa a cantidade de requisitos sen completar no Product Backlog ao inicio de cada xornada, permitindo ao equipo verificar se o ritmo é axeitado e predicir a finalización.

- **Definition of Done:** Conxunto de criterios e estándares comúnmente acordados polo equipo para determinar cando unha historia de usuario ou tarefa está completamente rematada. Inclúe aspectos como probas unitarias, documentación actualizada, implementación de funcionalidades solicitadas e validación do Product Owner, garantindo a calidade e entregabilidade.

- **Definition of ready:** Listaxe de requisitos e condicións previas que unha historia de usuario debe cumprir antes de iniciarse o seu desenvolvemento. Normalmente inclúe detalles sobre aceptación, deseño, criterios claros, e dependencia resoltas por actores externos ao equipo, adiantando posibles bloqueos e mellorando a planificación.

4.1.6 Scrum no proxecto

Para poder levar a cabo este proxecto dunha forma moito máis axeitada e cun equipo máis reducido se tomaron unha serie de decisións, os ”Product Owner” serán os directores a cargo deste traballo académico, Miguel Anxo Pérez Vila e Eliseo Bao Souto , o papel do resto do equipo, é decir, do Scrum Master e o desenvolvedor corre a cargo de Juan Toirán Freire. As reunión diarias deixan de ter sentido porque está todo desarrollado por unha soa persoa.

4.2 Planificación do proxecto

Para poder levar a cabo calquera tipo de proxecto sempre é necesario ter antes unha estimación dos custos, riscos e recursos necesarios, para o noso proxecto imos detallalos a continuación.

4.2.1 Recursos

Software

Utilízanse diferentes frameworks e programas para poder desarrollar a solución. Están explicados en detalle no capítulo CITARCAPITULO.

Hardware

Grazas ao Information Retrieval Lab (IRlab) tiven acceso tanto para facer o fine tuning como a inferencia co modelo a un servidor que conta cunha GPU NVIDIA RTX A6000 de 48GB e 128GB de RAM. Ademáis de iso utilízase o computador persoal do alumno que non ten ningún requisito especial, é un equipo básico sen requisitos mínimos

Humanos

por un lado están os tutores Miguel Anxo Pérez Vila e Eliseo Bao Souto que tendrán os roles de product owners e logo oalumno, Juan Toirán Freire, que asume tanto o de Scrum master e o de equipo de desenvolvemento a ademáis será necesaria unha persoa profesional no ámbito da ciberseguridade. Isto da a entender que o alumno será o encargado de analizar, deseñar e desenvolver o proxecto á vez que cumpre co marco metodolóxico. O experto en ciberseguridade non imputará coste porque é accesible mediante o grupo de investigación.

4.2.2 Custos

Os custos totais do proxecto derivan das horas empregadas polos diferentes traballadores e a infraestructura, sendo o custo principal do proxecto ás horas de traballo do persoal cualificado.

Posto	Custo	Tempo(h/sprint)	Sprints	Total
Directores	50 €/h	10	6	3000 €
Desenvolvedor	25 €/h	45	6	6 750 €
Analista	30 €/h	5	6	900 €
Experto en ciberseguridade	30 €/h			150 €
Total				10800 €

Tabla 4.1: Custos asociados ao capital humano.

Útil	Custo	Vida útil (meses)	Utilización	Total
Servidor	10000 €	60	5	833.33 €
Ordenador	1200 €	96	5	62.5 €
Total				895.83 €

Tabla 4.2: Custos asociados ao hardware.

Non ningún custo asociado ao software porque o que se utilizou era gratuito, os custos asociados ao experto en ciberseguridade non teñen reflexadas as horas de adicación na táboa REFERENCIA porque eran feedback progresivo que se ía obtendo ao largo do tempo cunha adicación aproximada de 5 horas. Ao gasto total do proxecto falta por engadir o gasto asociado aos suministros (electricidade e wifi) que poden contabilizarse aproximadamente ao longo dos 5 meses en 100 € dando un custo final total de 11795.83 €.

4.2.3 Riscos

Todo proxecto tecnolóxico conleva incertezas, e este proxecto non foi unha excepción. Como parte da xestión do proxecto, realizouse unha identificación e análise dos principais riscos potenciais, avaliando o seu impacto e probabilidade para determinar a exposición ao risco para cada caso. A continuación descríbense os principais riscos detectados:

- **Risco técnico de rendemento do LLM:**

  Existe o risco técnico de que o LLM finetuneado non alcance o rendemento agardado, tanto en calidade (por exemplo, xerando respostas imprecisas ou alucinacións perigosas) como en velocidade de inferencia (consultas demasiado lentas). Para mitigalo, desplegouse unha arquitectura RAG para reducir alucinacións e mellorar exactitude, realizáronse iteracións de fine tuning con datos de validación do dominio e, se fora necesario, plantexaríase cambiar a outro LLM ou reforzar o preprocesamento dos prompts. No aspecto computacional, optimizouse o despregue (GPU, lotes pequenos) e estúdianse técnicas de cuantización ou redución de modelo; no peor caso, limitaríanse opcións en tempo real ou ofreceríase respostas máis breves para manter a ferramenta operativa.

  - **Probabilidade:** Medio
  - **Impacto:** Alto
  - **Exposición:** Alto

- **Risco de dispoñibilidade e calidade de datos de ciberseguridade:**

  O proxecto depende en parte de contar con información relevante e ben estruturada, pero existe a posibilidade de non atopar datos suficientes ou de que estes sexan difíciles de procesar. Isto degradarían a especialización do modelo e a eficacia do RAG, malia existir moitas fontes abertas obter contido axeitado require dunha boa labor de busca e filtrado. Para mitigalo, planificouse dende o comezo unha fase de recollida e preparación dos datos, empregándose fontes recoñecidas e scraping de documentación oficial.

  - **Probabilidade:** Medio

  - **Impacto:** Medio
  - **Exposición:** Medio

- **Risco de integración de compoñentes:**

  Posibilidade de que, ao unir múltiples tecnoloxías, xurdan incompatibilidades ou erros que atrasen o desenvolvemento. Durante as fases iniciais era bastante probable atopar problemas no intercambio de datos, nas chamadas á API ou no despregue conxunto, o cal podía requirir depuración intensiva. Para mitigalo, optouse por un enfoque incremental: validar primeiro o fluxo LLM+LangChain con documentos sinxelos, engadir despois ElasticSearch como backend de busca e só logo construír a API, así se detectaron e resolveron cedo as incompatibilidades garantindo que ningún compoñente problemático bloquease o proxecto.

  - **Probabilidade:** Alto
  - **Impacto:** Medio
  - **Exposición:** Medio

- **Risco de cumprimento de prazos (xestión temporal):**

  Risco de non completar todas as funcionalidades do proxecto con data fixa de entrega, especialmente cando se empregan tecnoloxías emerxentes que complican a estimación temporal. O seu impacto é alto, xa que calquera atraso significativo pode comprometer a calidade e a presentación final. Para mitigar este risco adoptouse Scrum: estableceuse desde o inicio un MVP e priorizouse o backlog para garantir que o núcleo do proxecto quedase rematado con marxe; reavaliouse o progreso ao final de cada sprint para replanificar e axustar o alcance se había desvíos. Houbo en todo momento unha comunicación fluída cos titores para reaxustar expectativas; e, como último recurso, consideráronse horas extra para garantir os obxectivos críticos. O control continuo do burn-down chart permitiu chegar á data de entrega prevista co alcance acordado.

  - **Probabilidade:** Medio
  - **Impacto:** Alto
  - **Exposición:** Alto

- **Risco de seguridade e privacidade:**

  Existe a posibilidade de que actores maliciosos saquen proveito da ferramenta (por exemplo, mediante prompt injection para obter datos sensibles ou manipular respostas), así como o perigo de expoñer información confidencial ou non cumplir correcta-

mente todas as normativas de LOPD y RGPD. Este risco non é central no traballo pero sí importante telo en conta.

- **Probabilidade:** Baixo
- **Impacto:** Medio
- **Exposición:** Baixo

Capítulo 5

Extracción de datos

A fase de extracción de datos constitúe un piar esencial para asegurar que o chatbot de auditoría conteña información relevante, actualizada e de calidade no ámbito da ciberseguridade. Nesta sección describiranse detalladamente as dúas fontes principais empregadas —o foro de Gentoo e o foro de Debian— e os criterios de filtrado aplicados para recoller un conxunto de datos consistente, centrado exclusivamente en cuestións resoltas (etiquetadas cun "SOLVED" no título).

5.1 Fontes de datos

Para acadar un corpus de discusións e solucións no eido da ciberseguridade, escolléronse dous dos foros máis reputados e activos na comunidade open-source de Linux:

Figura 5.1: Logotipo de Debian.

- **Foro de Gentoo:** O foro oficial de Gentoo destaca por unha comunidade técnica moi especializada, na que profesionais e entusiastas participan activamente na resolución de problemas avanzados. Dentro deste espazo, identifícanse canles e subforos específicos destinados á seguridade —por exemplo, discusións sobre hardening de sistemas, configuración de SELinux ou estudo de vulnerabilidades en paquetes de software—. Para garantir a relevancia dos datos, realizouse unha extracción selectiva de fíos de discusión que conteñen no seu título o marcador "SOLVED", sinal inequívoco de que a pregunta formulada recibiu unha resposta satisfactoria que pechou o caso. Deste modo, o dataset

recolle non só o problema orixinal, senón tamén a solución final validada pola comunidade, proporcionando exemplos reais de diagnóstico, mitigación e boas prácticas.

Figura 5.2: Logotipo de Gentoo.

- **Foro de Debian:** O foro de Debian, igualmente, presenta un alto volume de interacións centradas en cuestións de seguridade, desde configuración de firewalls ata actualizacións seguras de paquetes e análise de incidentes. Tal como no caso de Gentoo, filtrouse exclusivamente o contido calificado como resolto mediante o sufixo "SOLVED" no título das publicacións. Esta etiqueta serve como criterio obxectivo para diferenciar as discusións que alcanzaron consenso sobre unha solución e evitar así a inclusión de temas pendentes ou especulativos. Ademais, a variedade de niveis de usuario en Debian —que inclúe administradores de sistemas corporativos e desenvolvedores de paquetes— enriquece o conxunto de datos con enfoques técnicos diversos.

En conxunto, estas dúas fontes proporcionan un volume significativo de documentos estruturados en forma de fíos de discusión, nos que se detalla tanto o problema plantexado como as respostas e comentarios sucesivos ata a resolución definitiva. Ao centrarse só nas publicacións "SOLVED", garantimos que o chatbot aprenda de casos completos, esbozando non só síntomas ou signos de actividade maliciosa, senón estratexias efectivas de mitigación e corrección. Esta aproximación asegura unha base sólida para o posterior preprocesado, indexación e fine-tuning do modelo, optimizando a súa capacidade para suxerir recomendacións contrastadas e aplicables en escenarios reais de auditoría de seguridade de maneira axeitada.

5.2 Deseño do web crawler

Para automatizar a recollida de fíos resoltos sobre ciberseguridade dos foros de Gentoo e Debian, deseñouse un web crawler baseado en Scrapy, un framework de Python optimizado para rastrexar sitios web e extraer datos estruturados. A continuación descríbese o funcionamento conceptual deste crawler, as políticas adoptadas para respectar as condicións de uso (TOS) dos foros e o tratamento de filtros e elementos duplicados.

5.2.1 Scrapy

A ferramenta elixida foi Scrapy por varias razóns clave:

- **Arquitectura modular:** Scrapy separa claramente os compoñentes de rastrexado (spiders), extracción (parsers), e almacenamento (pipelines), o que facilita a amplíabilidade e mantemento do proxecto así como a integración con tecnoloxías externas (ElasticSearch).
- **Xestión automática de solicitudes e respostas:** Scrapy emplea un sistema asíncrono (Twisted) para enviar solicitudes HTTP en paralelo, reducindo o tempo total de rastrexado sen saturar o servidor web.
- **Ferramentas integradas de control de velocidade:** Permite configurar atrasos entre peticións (DOWNLOAD_DELAY), tempos de espera máximos (DOWNLOAD_TIMEOUT) e axustes do User-Agent, fundamentais para non sobrecargar os foros e pasar desapercibidos como un navegador convencional. Esta parte resultou fundamental xa que a veces foi necesario limitar a velocidade porque en algún foro excedíamos o límite permitido (debian).
- **Soporte a regras avanzadas de filtrado** Mediante rexistros XPath e expresións regulares, Scrapy facilita a identificación selectiva só das publicacións marcadas como "SOLVED" no seu título, descartando todo o demais.
- **Integración con ElasticSearch:** A través de pipelines personalizadas, Scrapy pode enviar directamente os elementos extraídos a ElasticSearch, permitindo indexar de forma automática o contido para consultas de RAG e recuperación eficiente de información.

5.2.2 Políticas de crawling e cumprimento de TOS

Para asegurar un comportamento respectuoso e plenamente acorde cos termos de servizo dos sitios, primeiro analizouse o ficheiro robots.txt e comprobouse que as rutas /viewforum.php e /viewtopic.php non estaban restrinxidas. Como a extracción de contido público con fins académicos, sen redistribución masiva estaba permitida comezamos ca extracción destes datos públicos. Co obxectivo de evitar un posibel bloqueo do servidor e evitar ser tomado por tráfico malicioso, configurouse un cabeceiro User-Agent que simula un navegador real ("Mozilla/5.0 ... Chrome/123.0.0.0") pero conserva unha cadea distintiva que permite recoñecer o crawler.

Tamén co fin de protexernos das limitacións de acceso do foro de debian para este crawler limitouse a carga exercida mentres que en gentoo non era necesario. No primeiro caso agregouse un atraso fixo de dous segundos entre peticións e definiuse un timeout máximo de

sesenta segundos, co fin de impedir que solicitudes excesivamente lentas bloquearan o proceso de obtención de datos pero tamén intentar evitar que podan ser bloquedas por moi rápidas. Además de todas estas comprobacións, grazas a Scrapy volve intentar automaticamente as peticións que reciben códigos 5xx (por defecto o RetryMiddleware de Scrapy), e o seu sistema de logs.

5.2.3 Xestión de filtros

Para garantir un dataset limpo e libre de duplicados, implementouse un filtrado por marcador "SOLVED" no parser de títulos mediante expresións regulares que recoñecen variantes de "solved" (entre corchetes, parénteses ou soa) e exclúense as que conteñen "unsolved", de modo que só se seguen enlaces a fíos realmente resoltos; ademais, aproveitouse o dupefilter por defecto de Scrapy para evitar solicitar dúas veces a mesma URL, garantindo que cada tema se procesase unha única vez aínda que aparecese en diferentes páxinas ou con parámetros de paginación distintos; finalmente, aplicáronse regras de limpeza ao extraer o HTML de cada publicación para eliminar firmas, liñas de separación e metadatos irrelevantes e eliminar o propio código HTML quedando gardado só o texto puro das preguntas e respostas. Esto busca mellorar a fiabilidade das comprobacións posteriores de duplicidade e calidade dos datos para a obtención da resposta correcta dentro do dataset.

5.3 Preprocesado e limpeza

Nesta etapa, partimos dos ficheiros orixinais en formato JSON, un para Debian e outro para Gentoo e aplicamos un conxunto de operacións encamiñadas a homoxeneizar o contido textual, eliminar ruído e metadatos irrelevantes, e facilitar a anotación da resposta "correcta" co obxectivo de construir un dataset de fine-tuning de calidade. A continuación descríbense con detalle os pasos seguidos.

5.3.1 Obtención e análise de estatísticas iniciais

Para entender a diversidade e estructura das respostas nos foros, primeiro "aplanamos" cada JSON nun DataFrame de pandas onde cada fila representa un par cuestión–resposta.

Aplanado da estrutura JSON

Primeiramente limpanse os datasets de cadeas conexas que non están relacionadas e chegaron no proceso de extracción de datos e logo realizamos un aplanado da estrutura JSON, o obxectivo é transformar cada obxecto complexo coma un "tema" coas súas múltiples respostas a nunha táboa bidimensional na que cada fila corresponde a unha única resposta asociada á

súa pregunta orixinal. Para iso, lévase a cabo un percorrido de todos os elementos dos ficheiros json. Asegurámonos de non ter preguntas baleiras e despóis realiza de maneira axeita a asignación das distintas varaibles ás columnas como poden ser url, title, ask, timestamp_creation, answer_username, answer_rank e answer_text.

Inspección de valores únicos de rango

Tras a carga en DataFrame, imprímense os valores únicos da columna answer_rank para identificar os distintos niveis de autoridade presentes. Atopamos para os foros de gentoo e debian os seguintes rangos:

Debian	Gentoo
Debian Developer, Site Admin	Administrator
Debian Developer, Forum Ninja	Developer
Site admin	Retired Dev
Administrator	Moderator
Global Moderator	Arch/Herd Tester
Section Moderator	Watchman
Moderator Team Member	Guru
Debian Developer	Veteran
Emeritus	l33t
User Project Contributor	Bodhisattva
Forum Helper	Advocate
df -h | grep > 90TiB	Apprentice
df -h | grep > 20TiB	Tux's lil' helper
df -h | participant	n00b
Forum Account	

Personas sin rango asociado

Tabla 5.1: Diferentes rangos de los foros ordenados de mayor a menor impor

A parte de analizar todos os tipos diferentes de rangos dos foros ca finalidade de poder utilizalos a posteriori para catalogar as respostas correctas verificáronse outras estadísticas como os tipos de datos (asegurando ter eliminadas as preguntas sen resposta ou con erros de tipos)

5.3.2 Limpieza de ruído e contido irrelevante

Durante a fase de normalización, cada fragmento de texto (títulos, preguntas e respostas) sometese a un proceso sistemático que elimina todos os caracteres de control o espaciado excesivo: primeiro, sustituense todas as secuencias de salto de liña (`\n`, `\r\n`) e tabulación (`\t`) por espazos simples, co fin de evitar rupturas inesperadas no fluxo de texto; a continuación, aplícase unha expresión regular que agrupa tódalas ocurrencias de espazos, tabulacións ou saltos de liña consecutivos nun único espazo, garantizando unha separación uniforme entre palabras; por último, émplease o método strip() para retirar os espazos sobrantes ao principio e ao final da cadena. Este tratamiento, implementado produce un texto continuo, libre de sal-

tos redundantes, fundamental para asegurar a coherencia e a homoxeneidade dos datos antes das etapas posteriores de filtrado e selección de respostas.

5.3.3 Anotación e etiquetado da resposta correcta

Na fase de anotación e etiquetado da resposta correcta, para cada publicación seleccionamos, de entre todas as respostas, aquela que se considerará canónica baseándonos nun criterio obxectivo de autoridade e temporalidade. Primeiro establécese unha xerarquía de rangos de usuarios (dende administradores e desenvolvedores ata colaboradores ou xente sen rol) que reflicte o nivel de coñecemento e responsabilidade de cada quen. Logo, de todas as respostas publicadas antes de que o autor marcara o fío como resolto, valórase o rango de quen as escribiu e escóllese a última de todas do usuario con maior autoridade. Deste xeito garantimos que, para cada pregunta, o modelo reciba como exemplo de "completition" aquela resposta que non só responde correctamente, senón que o fai desde a perspectiva máis experimentada e fiable. Unha vez feita esta elección, constrúese finalmente o conxunto de pares pregunta–resposta, asegurando que o modelo só aprenda de datos limpos, relevantes e homoxéneos, o cal resulta esencial para un finetuning de calidade en contornas de ciberseguridade. Podemos ver na táboa (citar a taboa de arriba cando revise como) os ordes dos distintos rangos dos foros

5.4 Almacenamento e formato de datos

Neste apartado descríbese como se organizan, almacenan e expón os datos que conforman o noso dataset, así como as ferramentas elexidas para a súa consulta eficiente. A elección dunha estrutura clara e de ferramentas de busca potentes é fundamental para que o chatbot poida recuperar con rapidez a información relevante e xerar respostas precisas.

5.4.1 Esquema do dataset

O noso dataset está composto por documentos extraídos de foros de discusión (Debian e Gentoo) e publicados en Hugging Face nun formato JSON line-delimited. Cada rexistro representa unha conversa ou fío completo e inclúe un conxunto de metadatos e campos de contido, co seguinte esquema:

- **url:**
  - Ligazón orixinal ao fío do foro.
  - **Tipo:** *keyword*
  - *Exemplo:* "https://forums.debian.net/viewtopic.php?t=161929"
- **title:**

  - Título da entrada principal, normalmente coa etiqueta "[Solved]" cando o problema foi resolto.
  - **Tipo:** *text*
  - *Exemplo:* "[Solved] Eduroam WiFi not connecting"

- **ask:**
  - Texto completo da pregunta ou descrición do problema formulado polo usuario orixinal.
  - **Tipo:** *text* con sub-análise
  - *Exemplo:* "Hi, I'm usually connected to an EDUROAM wifi network. Up to a few days ago, it was working just fine..."

- **timestamp_creation:**
  - Data e hora (en formato ISO 8601) da publicación orixinal.
  - **Tipo:** *date*
  - *Exemplo:* 2025-04-03T22:05:02+00:00

- **timestamp_edit:**
  - Data e hora da última edición ou actualización da entrada principal, se procede.
  - **Tipo:** *date*
  - *Exemplo:* "2025-04-10 12:26"

- **correct_answer:**
  - Texto da resposta marcada como correcta ou aceptada polo autor do fío.
  - **Tipo:** *text* con un sub-análise
  - *Exemplo:* "Right, made some progress! I have discovered this happens due to NAT being enabled. This section is present in my server's /etc/nftables.conf..."

- **other_information:**
  - Obxecto libre, actualmente usado para almacenar calquera atributo adicional (por exemplo etiquetas, votos, categorías) que en principio non se definiron formalmente.
  - **Tipo:** *object*

- **answers:**

  - **username** (*keyword*): nome de usuario do respondente.
  - **rank** (*keyword*): rangos especiais (por exemplo "Global Moderator") ou cadea baleira.
  - **published_at** (*date*): data de publicación da resposta.
  - **response_text** (*text*): contido textualmente completo da resposta.

Este deseño modular permite, por un lado, indexar de forma separada e eficiente cada bloque de texto suxeito a buscas por palabra ou frase, e por outro, filtrar ou agregar por metadatos (fecha, usuario, prioridade, etc.).

5.5 Uso de ElasticSearch

Para facilitar unha busca rápida e flexible sobre estes rexistros estruturados en JSON, utilizamos ElasticSearch como motor de indexación e consulta. ElasticSearch ofrécenos unha serie de vantaxes que son:

- **Indexación de documentos JSON:**
  - Campos text (ask, title, response_text, correct_answer) configurados con analizadores de linguaxe (tokenización, eliminación de stop-words) para soporte de buscas full-text.
  - Campos keyword (url, username, rank) para filtrado exacto e agregacións.
  - Campos date para filtrado por rango temporal e ordenación cronolóxica.
- **Consultas full-text enriquecidas:**
  - **Match queries** para atopar documentos que coincidan con termos ou frases no contido.
  - **Bool queries**que combinen criterios de texto, metadatos e rangos de datas.
  - **Highlighting**, para resaltar as partes relevantes do texto que satisfacen a consulta.
- **Filtrado e agregación:** podendo filtrar por rango de datas, agrupar por usuarios ou facendo faceting por etiquetas.
- **Rendemento e escalabilidade:** permitindo distribuir índices nunha ou varias máquinas permitindo crecemento horizontal e alta dispoñibilidade, ademáis de consultas moi rápidas con moitos documentos grazas a estrutura de almacenamento invertido e ao almacenamento interno de vectores cando empregamos índices densos (embbedings).

- **Integración co pipeline RAG:** Na nosa arquitectura de Retrieval-Augmented Generation (RAG), as chamadas de recuperación formulan directamente consultas a ElasticSearch para extraer os fragmentos ou documentos máis relevantes, que logo se achegan como contexto ao modelo de linguaxe para xerar respostas máis precisas e fundamentadas. Grazas á capacidade de reindexar incrementalmente, o índice tería a posibilidade de manterse ao día garantindo que o chatbot dispoña en todo momento da información máis recente nun futuro.

Capítulo 6

Creación da web

Este capítulo céntrase en presentar o deseño detallado e a implementación dos compoñentes que conforman o sistema, dividindo o contido a grandes rasgos en dúas áreas: a interface web (frontend) e a arquitectura técnica subxacente (backend e servizos asociados). O seu obxectivo é explicar como se constrúe a experiencia de usuario a través da aplicación en React, permitindo interaccións dinámicas cun chatbot especializado en ciberseguridade. A continuación, describiremos de forma pormenorizada a comunicación co servidor, a lóxica de procesamento e recuperación de información, así como a integración co modelo de linguaxe grande (LLM) despregado nun servidor mediante FastAPI. Tamén explicaremos como o backend emprega un pipeline RAG con Elasticsearch para recuperar documentos relevantes e, despois, xera as respostas co modelo LLM.

6.1 Arquitectura xeral da aplicación e fluxo entre compoñentes

6.2 Frontend (React)

A interface gráfica está construída empregando React, decidiuse utilizar esta tecnoloxía xunto con Django aproveitando o mellor de ambos frameworks, concretamente de React aproveitamos a versatilidade que nos ofrece á hora de facer interfaces dinámicas, con boa UX/UI, cunha múltiple cantidade de plantillas e ampla comunidade ademáis da simplicidade e eficiencia na creación de compoñentes reutilizables e interactivos. En lugar de usar clases, utilizáronse compoñentes funcionais de React xunto cos Hooks (useState, useEffect, etc.) para manexar o estado interno e os efectos secundarios da aplicación. Por exemplo, mantense en estado o texto da consulta introducida polo usuario e a lista de mensaxes da conversa, de modo que a interface se actualiza de forma reactiva a cada nova pregunta ou resposta.

A comunicación co backend realízase mediante chamadas API REST dende o propio React. Cando o usuario introduce unha pregunta e pulsa en enviar, o frontend realiza unha petición

HTTP ao servidor Django expoñendo un endpoint REST. Esta integración permitiu separar claramente a lóxica de negocio (no backend) da presentación (no cliente). Ademais, configurouse CORS no backend para permitir que a aplicación React (normalmente servida en localhost:3000 durante o desenvolvemento) poida realizar peticións ao servidor Django (por exemplo en localhost:8000) sen erros. Unha vez que o backend devolve a resposta, React actualiza o estado correspondente (engadindo a resposta do chatbot á conversa) e re-renderiza os compoñentes para mostrar o novo intercambio ao usuario.

Desde o punto de vista de UX, a aplicación React implementa unha interface de chat: un campo de entrada de texto para a consulta, un botón de envío, e un panel onde se listan de forma cronolóxica as mensaxes do usuario e as respostas do asistente. Cada vez que chega unha resposta do backend, engádese unha novo burbulla de texto no panel de chat. Para mellorar a experiencia, manéxanse estados de carga (por exemplo, mostrando un indicador mentres se espera pola resposta) e erros (notificando ao usuario se houbo algún problema na comunicación co servidor).

Para a pantalla de chat de React escolleuse unha plantilla coñecida polo seu deseño moderno e adaptativo, denominada NOME E LINKPLANTILLADECHAT. Esta plantilla ofrécenos unha estrutura visual limpa, cun espazo reservado para a lista de conversas na parte lateral e unha área principal onde se mostran as burbullas de mensaxe, perfectamente deseñadas para garantir unha lectura cómoda e unha vista pouco recargada. Nesta pantalla todas as seccións do chat xa veñen configuradas con estilos responsive, espazamento adecuado entre elementos e tipografías que facilitan a distinción entre mensaxes do usuario e respostas do asistente. Ademais, esta plantilla integra de forma natural iconos para as accións (enviar, anexar ficheiros, editar), o que mellora a coherencia visual e reduce o esforzo necesario para axustar detalles de estilo dende cero.

No caso dos formularios, empregouse a plantilla LINKPLANTILLADEFORMULARIO, é versatilidade para xestionar entradas de texto, selección de opcións e validación visual de campos obrigatorios. NOMEPLANTILLA LINKPLANTILLADEFORMULARIO inclúe xa un deseño de etiquetas, campos de texto e botóns con estados de foco, erro e éxito, permitindo unha experiencia fluída á hora de achegar datos ao sistema. Ao reutilizar esta plantilla, logrouse uniformidade nos formularios de rexistro, edición de perfil e creación de novos chats, sen ter que reinventar a roda (REFERENCIAR PATRÓN reinventing the wheel). Deste xeito, o usuario percibe un estilo consistente en todo o percorrido, desde que accede á páxina principal ata que envía unha mensaxe ou modifica o título dun chat.

Ambas plantillas, LINKPLANTILLADECHAT e LINKPLANTILLADEFORMULARIO, destacaron pola facilidade para adaptar as cores corporativas e os tamaños de tipografía, polo que se puideron integrar perfectamente coas directrices de estilo do proxecto sen necesidade

de grandes cambios. Tamén ofrecen unha disposición optimizada para dispositivos móbiles, asegurando que tanto o chat como os formularios se vexan correctamente en pantallas pequenas, con botóns de suficiente tamaño e campos de texto adaptados ao táctil. Desta maneira, garantíase unha boa accesibilidade desde o inicio do desenvolvemento, aforrando tempo e mellorando o resultado final de cara á experiencia dos usuarios.

6.3 Backend (Django)

No lado servidor, o framework Django actúa como núcleo do backend da aplicación. Django encárgase de recibir as peticións REST enviadas dende o frontend React, xestionar a lóxica da aplicación e coordinar os distintos compoñentes internos para xerar a resposta do chatbot. Neste proxecto, Django serve basicamente como unha API que implementa o patrón RAG (Retrieval-Augmented Generation) para responder as consultas dos usuarios en base a documentación local.

Ao empregar Django, beneficiámonos da súa robustez e do ecosistema de librarías en Python. En particular, integrouse Django REST Framework para facilitar a creación de endpoints REST e xestionar autenticación e os permisos se fosen necesarios. Cando React realiza unha petición (por exemplo, un POST cunha pregunta), Django pasa esa petición a unha vista (ou ViewSet de DRF) que contén a lóxica do chatbot. Dentro desta lóxica, o backend accede aos datos locais dispoñibles – concretamente un conxunto de logs e textos extraídos de foros de ciberseguridade que forman a base de coñecemento do sistema. Estes datos foron procesados e almacenados previamente para permitir a busca semántica.

Unha vez recibida a pregunta, o backend Django coordina os seguintes pasos:

- **1.** Utiliza a integración con Elasticsearch para recuperar dos datos locais aqueles documentos máis relevantes para a consulta.
- **2.** Constrúe un prompt que combina a pregunta do usuario co contexto informativo recuperado.

Toda esta funcionalidade apóiase en LangChain, xa que esta libraría facilita o encadeamento de pasos en aplicacións impulsadas por modelos de linguaxe. Tanto Elasticsearch como LangChain están integrados no backend e execútanse localmente, o que significa que as búsquedas de información e a preparación do contexto lévanse a cabo dentro da infraestrutura local, sen depender de servizos externos na nube (como si que pasa no caso do LLM). Isto garante menor latencia e maior control sobre os datos.

Podemos ver a parte de Django como un orquestrador: recibe a consulta de React, accede á base de coñecemento local, prepara o prompt enriquecido coa información relevante, e en-

cárgase de invocar o modelo de linguaxe para obter unha resposta. Unha vez obtida a resposta do modelo, Django envía esta resposta de volta ao frontend en formato JSON, para que React poida mostralo ao usuario. Esta separación de responsabilidades segue unha arquitectura limpa e escalable (REVISAR DOCU DE PATRÓNS POR SE PODO METER ALGÚN), permitindo que o frontend permaneza simple (solicitando e mostrando información) mentres o backend asume todo o peso da comprensión e busca da información.

6.4 Pipeline RAG no backend con LangChain e Elasticsearch

No núcleo do sistema atópase unha implementación local dunha pipeline de Retrieval-Augmented Generation (RAG), distribuída entre Django (recuperación) e o servidor de modelo (xeración). Esta pipeline segue as mellores prácticas para combinar busca de información e xeración por IA, garantindo que as respostas do chatbot se basen no coñecemento dispoñible nos datos locais.

Dentro do backend Django, LangChain intégrase con Elasticsearch para realizar a fase de retrieval. Cando chega unha nova consulta do usuario, primeiro xérase a representación vectorial (embedding) da consulta empregando un modelo de embedding semántico. Este modelo de embedding está adestrado para proxectar textos do dominio de ciberseguridade a un espazo vectorial de altas dimensións, de forma que textos semellantes queden a distancia curta nese espazo. A libraría LangChain facilita este paso: por exemplo, empregando unha clase de Embeddings pre-configurada (pódese utilizar un modelo tipo SentenceTransformer ou incluso o propio LLaMA nun modo especial de xeración de embeddings). Obtéñense así uns vectores numéricos que representan tanto a pregunta do usuario coma os documentos do foro/logs almacenados.

Unha vez calculado o vector da consulta, realízase unha busca semántica en Elasticsearch para atopar os documentos máis relevantes. Elasticsearch está configurado como un vector store que almacena embeddings dos documentos; a consulta vectorial devolve os N documentos ou fragmentos con maior similitude co embedding da pregunta. Nesta fase, úsase a funcionalidade de k-NN search de Elasticsearch para comparar o vector da pregunta cos vectores indexados e recuperar, os fragmentos que mellor coinciden co contido da pregunta. Este enfoque de busca semántica permite atopar información relevante aínda que non haxa coincidencia literal de palabras, aproveitando a semellanza de significado.

Coas pasaxes relevantes na man, o backend prepara o contexto para o modelo de linguaxe. Normalízanse e formateanse os textos recuperados e combinandose nun único prompt xunto coa pregunta orixinal do usuario. Habitualmente emprégase un modelo de prompting do tipo:

"A continuación móstranse algúns documentos relevantes da base de coñecemento. Responde á

pregunta do usuario empregando só a información achegada: [documentos contextuais] Pregunta: [pregunta do usuario]".

Este prompt composto é o que constitue a parte "retrieval-augmented" da pipeline RAG: engade datos externos (retrieved docs) ao prompt do modelo para guialo.

É importante salientar a separación entre a etapa de recuperación e a etapa de xeración no deseño RAG. A recuperación de información relevante efectúase integramente no backend local (Django+Elasticsearch), mentres que a xeración da resposta está delegada ao modelo de linguaxe grande (LLM) externo. Deste xeito, o LLM non ten que "memorizar" todo o corpus de datos locais, senón que recibe en cada consulta só un subconxunto relevante de información. Este enfoque mellora tanto a pertinencia das respostas (o LLM dispón de contexto específico para a pregunta) como a eficiencia e escalabilidade do sistema (pódese actualizar a base de coñecemento en Elasticsearch sen necesidade de re-adestrar o modelo principal). Unha vez formado o prompt con contexto, o backend pasa á seguinte fase: a comunicación co modelo de IA a través da API.

6.5 Xestión dos embeddings e indexación en Elasticsearch

Un compoñente fundamental da arquitectura é a base de coñecemento vectorial construída enriba de Elasticsearch. Antes de poder responder preguntas, foi necesario preprocesar os datos locais e cargalos nun índice de Elasticsearch coa súa representación en forma de vectores (embeddings). A continuación detállase como se xestionan estes embeddings e a indexación:

Obtención e almacenamento de embeddings: Os documentos orixinais foron inicialmente normalizados e divididos se eran moi extensos, asegurando que cada fragmento teña un tamaño manexable para o modelo, despois aplicouse un modelo de embedding para converter cada fragmento de texto nun vector de dimensión fixa. Este proceso xera unha representación numérica do contido semántico de cada documento; textos semellantes xeran vectores próximos entre si no espazo vectorial. Os vectores resultantes, xunto cun identificador e posiblemente meta-datos do documento, foron almacenados en Elasticsearch aproveitando o tipo de dato especial dense_vector. Configurouse o índice de Elasticsearch de maneira que cada documento ten un campo vectorial (por exemplo, embedding) de dimensión d (a dimensión específica do modelo de embedding utilizado). Este campo está indexado para permitir buscas por similitude e definíronse os parámetros de comparación axeitados.

Métricas de similitude e normalización: Elasticsearch ofrece diferentes métricas para medir a semellanza entre vectores, sendo as máis comúns a similitude de coseno e o produto escalar (dot product).

- **A similitude de coseno** é frecuentemente a opción por defecto en Elasticsearch e

resulta útil cando os vectores embedding non están normalizados de orixe, xa que o cómputo do coseno efectivamente ten en conta a magnitude.

- **O produto escalar** pola súa banda, pode ser máis eficiente pero require que os vectores estean previamente normalizados (é dicir, escalados a magnitude 1) para que o resultado do dot product sexa equivalente á semellanza de coseno

No índice, os vectores foron normalizados á hora de introducilo, de forma que se podería empregar produto escalar como métrica con eficiencia. Independentemente da métrica elixida, o importante é que o índice vectorial permite busca por similitude: dado un novo vector de consulta, Elasticsearch devolve os documentos cuxos embeddings teñen maior proximidade (os máis relevantes semanticamente).

Pasos que se están a seguir:

- Chega unha pregunta do usuario e xérase o seu embedding vectorial.
- Emprégase a API de k-NN search de Elasticsearch (ou a interface proporcionada por LangChain) para obter os k vectores do índice máis próximos ao vector da pregunta
- Elasticsearch calcula as puntuacións de similitude (coseno ou dot product) entre o vector de consulta e cada vector almacenado
- Retornanse as mellores coincidencias ordenadas segundo a similitude.

Grazas a esta indexación previa, a busca é moi rápida incluso con miles de documentos. Cabe destacar que o uso de Elasticsearch permite tamén escalar a un conxunto de datos maior, aproveitando as capacidades de sharding/replicación do motor de busca se fose necesario. A xestión de embeddings e a súa indexación implica converter toda a base de coñecemento de ciberseguridade a representacións vectoriais e almacenalas en Elasticsearch para posibilitar unha recuperación semántica eficiente. Este almacenamento vectorial local proporciona a información de contexto que alimenta o modelo de IA, asegurando que as respostas estean baseadas en feitos ou recomendacións extraídas das fontes orixinais en lugar de depender unicamente do coñecemento interno do modelo. O resultado é un chatbot capaz de dar respostas especializadas e actualizadas ao dominio da ciberseguridade dun xeito rápido e preciso.

6.6 Conexión co modelo remoto vía FastAPI

A xeración da resposta lévase a cabo mediante un modelo de linguaxe LLaMA 3.1 de 8.000 millóns de parámetros, especializado en ciberseguridade, que, debido ás súas altas esixencias computacionais, está despregado nun servidor independente accesible a través dunha

API REST desenvolvida con FastAPI, para servir modelos de Machine Learning en produción. Neste esquema, o servidor remoto carga o modelo e ofrece un endpoint ao que o backend en Django envía, mediante unha petición HTTP o prompt preparado co contexto e a pregunta do usuario. O servidor FastAPI procesa este prompt co modelo LLaMA, que realiza a inferencia pasando a entrada polas súas capas neuronais para xerar unha resposta en linguaxe natural, que logo devolve tamén en formato JSON. Django recibe esta resposta, extrae a información relevante e envíaa ao frontend React, todo isto nun proceso rápido e transparente para o usuario final, que simplemente percibe que o chatbot respondeu á súa consulta. Esta arquitectura distribuída actualizar cada compoñente por separado, por exemplo, despregando o modelo LLaMA en máquinas con GPUs ou substituíndoo por outro máis avanzado no futuro, sen afectar o resto do sistema.

Capítulo 7

Entrenamento do modelo e xeración de embeddings

Neste capítulo detallarase todo o relacionado co Llama 3.1 8B escollido para a realización do proxecto, unha breve explicación dos datos utilizados, o método preciso para o axuste fino e todas as partes relevantes deste proceso.

7.1 Selección e configuración do modelo base

O modelo base seleccionado para o chatbot guardIAn foi LLaMA 3.1 con 8.000 millóns de parámetros (8B). Esta elección débese a varios factores. En primeiro lugar, LLaMA 3.1 é un modelo open-source de última xeración publicado por Meta en xullo de 2024. Ao ser de código aberto, os seus pesos e arquitectura están dispoñibles para uso e adaptación sen restricións propietarias, algo fundamental nun proxecto académico que busca flexibilidade e custo reducido. Ademais, a variante de 8B parámetros ofrece un equilibrio óptimo entre rendemento e consumo de recursos: con aproximadamente 16GB de VRAM requiridos para inferencia, este modelo pode executarse en GPUs máis lixeiras, evitando a necesidade de hardware de alta gama necesario para modelos de maior tamaño (70B ou 405B) que presentan esixencias moito máis altas (da orde de decenas ou centos de GB de memoria) e, aínda que ofrecen maior rendemento bruto, resultan inviábeis de despregar cos recursos dispoñibles.

Outro motivo para escoller LLaMA 3.1 8B é que incorpora melloras arquitecturais e de funcionalidade recentes. Todas as variantes de LLaMA 3.1 soportan un longo contexto de ata 128.000 tokens e son multilingües en oito linguas, incluíndo o inglés e o español (que son os idiomas para os que está pensado principalmente o proxecto). Unha maior lonxitude de contexto resulta útil no dominio da ciberseguridade, onde é frecuente analizar arquivos de log extensos ou configuracións longas. Igualmente, o soporte multilingüe permite que o chatbot entenda consultas e documentación en distintos idiomas técnicos. En conxunto, LLaMA 3.1

8B proporcionou unha base moderna, escalábel e alineada coas necesidades do proxecto: un modelo recente, de alto rendemento relativo, pero o suficientemente lixeiro para ser adestrado e despregado nun entorno universitario con recursos limitados.

7.2 Preparación do dataset

Para especializar o modelo no dominio de ciberseguridade, construíuse un conxunto de datos específico a partir de fontes de información técnicas. En particular, recompiláronse parellas de pregunta-resposta procedentes dos foros das distribucións Linux Gentoo e Debian. Estes foros conteñen discusións reais de usuarios e expertos sobre problemas de seguridade informática, configuración de sistemas, análise de rexistros (logs), entre outros temas afíns. A selección das parellas de pregunta e resposta realizouse seguindo criterios de calidade definidos no capítulo anterior PONERREFERENCIA, de xeito que só se incluíron exemplos relevantes para as tarefas obxectivo do chatbot (por exemplo, erro nunha configuración, resolución de erros de permisos, recomendacións de endurecemento de configuración, etc.). Priorizáronse aquelas cuestións con respostas aceptadas ou consensuadas pola comunidade, garantindo que o dataset conteña solucións correctas e ben explicadas.

O resultado foi un conxunto de datos de alta calidade, con milleiros de exemplos reais, que reflicten situacións coas que un profesional de ciberseguridade ou administrador de sistemas se atopa no día a día. Empregar datos reais e especializados en ciberseguridade é crucial por varias razóns. En primeiro lugar, asegura que o modelo aprende a terminoloxía do dominio (servizos, protocolos, mensaxes de erro, etc.) e como se expresan as preguntas e solucións neste ámbito. En segundo lugar, ao tratarse de problemas auténticos, evítase en parte o risco de alucinacións ou respostas irreais: o modelo tenderá a replicar solucións verificadas en lugar de inventar contido. Finalmente, un dataset tan enfocado proporciona ao LLM información detallada sobre nichos específicos (por exemplo, configuración de firewalls en Gentoo ou xestión de usuarios en Debian) que dificilmente estarían cubertos con profundidade suficiente no adestramento xeral dun LLM non especializado. Esta preparación do conxunto de datos sentou as bases para un axuste fino exitoso do modelo ao dominio de ciberseguridade.

7.3 Fine-tuning con LoRA

Unha vez dispoñible o dataset especializado, procedeuse a realizar o fine-tuning do modelo base para adaptalo ao dominio de ciberseguridade. O proceso de fine-tuning consiste en adestrar o LLM preadestrado co novo conxunto de datos de preguntas e respostas, de modo que axuste os seus parámetros á nova tarefa. Deste xeito, un modelo xeral como LLaMA 3.1 pode especializarse para proporcionar respostas máis precisas e contextualizadas sobre ciber-

seguridade. Porén, o adestramento completo dun modelo de 8B parámetros resultaría custoso en termos computacionais e podería provocar overfitting ou esquecemento do coñecemento xeral previamente adquirido polo modelo. Para abordar isto, empregouse a técnica LoRA (Low-Rank Adaptation) durante o axuste fino.

LoRA é unha técnica de adestramento eficiente de modelos grandes que permite axustar só un subconxunto pequeno de parámetros en lugar de todos os parámetros do modelo. En concreto, LoRA introduce matrices adicionais de baixo rango nas capas do modelo (por exemplo, nas proxeccións de autoatención) e conxela os pesos orixinais. Durante o adestramento, só se actualizan estas matrices reducidas, que posteriormente se combinan cos pesos principais para influír nas predicións. As principais vantaxes de empregar LoRA son:

- **Adestramento máis eficiente:** Ao concentrar o axuste nun número reducido de parámetros, diminúe o tempo de adestramento necesario, así como os recursos computacionais empregados. No noso caso, foi posíbel realizar o fine-tuning completo empregando unha única GPU, incluso aproveitando técnicas de cuantización para aforrar memoria, grazas a que LoRA reduce drasticamente o número de parámetros que hai que actualizar.
- **Preservación do coñecemento orixinal:** Dado que a maioría dos pesos do modelo base mantéñense conxelados, minimízase o risco de esquecemento podendo chegar a ser catastrófico. Un dos problemas do fine-tuning completo é que, ao alterar todos os pesos, o modelo pode desaprender información útil adquirida previamente. LoRA reduce esa posibilidade ao manter inalterada a gran parte dos pesos, conservando así o coñecemento xeral do modelo.

Grazas a LoRA, o proceso de fine-tuning foi moito máis lixeiro e rápido, permitindo iterar na mellora do modelo con recursos limitados. O resultado deste paso foi unha versión especializada de LLaMA 3.1 8B, afinada co noso dataset, á que nos referimos como modelo guardIAn. Este modelo ten agora a capacidade de xerar respostas seguindo o estilo e contido das solucións reais extraídas dos foros, combinando a potencia lingüística dun LLM xeral coa experiencia propia do dominio de ciberseguridade.

7.4 Xeración de embeddings

Un compoñente crucial da arquitectura do proxecto é o sistema de recuperación de información baseado en embeddings. Antes de responder a unha pregunta, o chatbot consulta unha base de coñecemento especializada (conformada polas fontes de datos de ciberseguridade, como os foros mencionados) para obter información relevante. Para habilitar isto, primeiro foi

necesario transformar os datos textuais (preguntas e respostas do corpus) nun formato axeitado para a busca semántica: xeráronse embeddings ou representacións vectoriais densas de cada documento. Un embedding é un vector de dimensión elevada (tipicamente de centos de compoñentes) que codifica o significado do texto; textos con contido semellante ou relacionado teñen vectores próximos no espazo vectorial. No noso deseño, empregouse un modelo de embedding preadestrado para converter cada documento (por exemplo, un fío de foro coa pregunta e a súa mellor resposta) nun vector numérico.

Unha vez obtidos os embeddings de todos os documentos do noso corpus de ciberseguridade, estes foron indexados nun servizo Elasticsearch configurado para busca vectorial. Elasticsearch permite almacenar vectores densos a través do tipo de campo especial "dense_vector", deseñado especificamente para gardar este tipo de representacións numéricas. Cada documento da base de coñecemento almacénase cun campo vectorial (co seu embedding) e, adicionalmente, con campos tradicionais (como texto completo, título, palabras chave, etc.). Ademais, o índice configurouse cunha métrica de similaridade apropiada para poder realizar consultas eficientes. Isto significa que, cando se proporciona un novo vector de consulta, Elasticsearch pode recuperar rapidamente os documentos cuxos embeddings sexan máis próximos a el, é dicir, os documentos máis semellantes en contido á consulta do usuario.

Este proceso de xeración e indexación de embeddings resulta fundamental para o proxecto. En lugar de depender unicamente do coñecemento almacenado nos parámetros do modelo (que pode estar desactualizado ou ser incompleto en temas moi específicos), o chatbot pode buscar en tempo real información pertinente nas fontes externas de ciberseguridade. Os embeddings actúan como ponte entre a linguaxe natural das consultas e os datos técnicos: permiten que unha pregunta do usuario atope, mediante proximidade semántica, as discusións ou guías nos foros que conteñen a resposta ou contexto apropiado. Deste xeito, asegúrase que o modelo traballe cun conxunto de datos actualizado e fiábel, mellorando a precisión e a fiabilidade das respostas xeradas.

7.5 Conexión coa pipeline RAG

Integramos os compoñentes previos empregando unha arquitectura de Retrieval-Augmented Generation (RAG), coa axuda do framework LangChain e do motor de busca Elasticsearch. O obxectivo dunha pipeline RAG é combinar un sistema de recuperación de información cun modelo xerativo, de forma que este último xere respostas fundamentadas en coñecemento recuperado dinámicamente. Na práctica, o fluxo de funcionamento é o seguinte:

- **Consulta do usuario:** O usuario introduce unha pregunta ou problema de ciberseguridade na interface do chatbot.

- **Busca de información:** Antes de xerar a resposta, o sistema crea un embedding da consulta do usuario (utilizando o mesmo modelo de embedding empregado para indexar o corpus). Con ese vector, realiza unha consulta no índice de Elasticsearch, obtendo os documentos máis semellantes. Por exemplo, se o usuario pregunta "Como podo arranxarr esta configuración de rede?", o sistema recuperará automaticamente fíos de foro ou artigos onde se discuta configuracións de rede erradas e a solución.

- **Combinación de contexto:** Os fragmentos de texto ou respostas relevantes atopados engádense como parte do contexto proporcionado ao modelo LLM. Empregando LangChain, establécese unha cadea de QA conversacional con recuperación: o prompt pasado ao modelo inclúe agora, ademais da pregunta do usuario, a información contextual recuperada (por exemplo, unha listaxe de pasos recomendados nun dos foros para ese problema).

- **Xeración da resposta:** O modelo (LLaMA 3.1 8B especializado) produce entón a súa resposta final, aproveitando tanto o seu coñecemento interno adestrado co fine-tuning coma os datos específicos proporcionados no contexto. Deste modo, a resposta xerada está fundamentada nas fontes de información reais, incorporando detalles ou solucións exactas mencionadas nos documentos relevantes.

- **Entrega ao usuario:** Finalmente, a resposta elaborada polo modelo entrégase ao usuario a través da interface web, que a mostra no formato de chat adecuado.

A integración de LangChain simplificou significativamente a implementación desta pipeline RAG. En particular, empregáronse as abstraccións de VectorStore de LangChain para conectar con Elasticsearch e as cadeas prefabricadas de Conversational Retrieval QA, que xestionan automaticamente os pasos de busca e inxección de contexto antes da xeración. Isto permitiu centrar os esforzos en axustar detalles (por exemplo, cantos documentos recuperar ou como formatear o prompt con contexto) sen ter que codificar todo o pipeline desde cero.

A arquitectura RAG dota ao chatbot de acceso a información fiábel dun modo que os LLMs illados non poden acadar. Mesmo un modelo ben adestrado pode fallar se se lle pregunta por unha vulnerabilidade descuberta despois do seu adestramento, ou por datos de configuración moi específicos; pola contra, no proxecto somos capaces de consultar a base de coñecemento proporcionada lembrando eses detalles no momento preciso. Isto reduce drasticamente a tendencia a alucinacións e respostas incorrectas, xa que o modelo ten unha base real na que apoiar as súas afirmacións. En termos xerais, RAG axuda a resolver este problema ao anclar o coñecemento paramétrico dun modelo xerativo cunha fonte externa de información, pasando esa información como contexto adicional e axudando así a xerar respostas máis relevantes

e precisas. Este enfoque híbrido demostrou ser especialmente eficaz en ciberseguridade, onde a exactitude da información e a constante actualización (novos parches, CVEs, técnicas emerxentes) son clave.

7.6 Despregue e inferencia

Unha vez adestrado e validado o modelo, procedeuse ao seu despregue nun entorno de servidor para permitir a súa utilización polos usuarios finais a través da aplicación web. O modelo especializado foi instalado nun servidor da universidade dotado de GPU, garantindo a dispoñibilidade dos recursos de cómputo necesarios para a inferencia. Para expoñer o servizo de forma accesíbel, implementouse unha API REST lixeira empregando FastAPI. Esta API ofrece un punto de entrada ao que a interface web pode enviar as consultas dos usuarios.

A decisión de utilizar FastAPI obedeceu a varias razóns: por unha banda, a súa sinxeleza e alto rendemento para crear servizos asincrónicos encaixa ben coa necesidade de atender peticións de inferencia que poden tardar varios segundos (especialmente cando o modelo xera respostas longas). Por outra banda, permite separar claramente o front-end do back-end do sistema sendo moi modular. No noso despregue, a interface de usuario web foi desenvolta en React mentres que o backend que recibe as respostan é en Django como parte do portal da aplicación, pero esa capa cliente non executa o modelo directamente; en lugar diso, cada pregunta ingresada polo usuario envíase á API de FastAPI, que actúa como ponte entre o frontend/backend da web e o modelo de IA no backend.

O proceso completo de inferencia despregado funciona así: o usuario interactúa co chatbot mediante a páxina web nunha ventá de chat. Cando preme en enviar a súa pregunta, o navegador remite esa consulta ao servidor Django, o cal á súa vez chama á API REST do modelo. A API recibe a pregunta e realiza internamente todo o pipeline descrito anteriormente (xeración do embedding, busca en Elasticsearch, construción do prompt con contexto e chamada ao modelo LLM para obter a resposta). Unha vez xerada, a resposta devólvese ao servidor web, que a procesa e preséntalla ao usuario na interface de chat. Este deseño desacoplado facilita a escalabilidade e o mantemento: o modelo LLM pode executarse illadamente no servidor con acceso dedicado á GPU, mentres que o servidor web maneja múltiples usuarios simultáneos, autenticación, presentación de resultados, etc. Ademais, permite futuras extensións a outros clientes (por exemplo, unha ferramenta de liña de comandos ou un bot nunha plataforma de mensaxería) simplemente consumindo a API REST, sen ter que duplicar a lóxica do negocio. En resumo, o despregue logrouse integrando as distintas pezas nun sistema coherente: un modelo LLM de código aberto afinado para ciberseguridade, enriquecido cunha base de coñecemento accesíbel vía embeddings, organizado mediante unha pipeline RAG, e servido a través dunha API robusta. Todo isto permitiu ofrecer aos usuarios unha ferramenta

innovadora e efectiva, froito dun enfoque clásico de enxeñaría potenciado polas técnicas máis modernas en intelixencia artificial aplicada ao dominio da seguridade informática.

Capítulo 8

Desenvolvemento do proxecto

Ao longo do proxecto identificáronse diferentes requisitos e historias de usuario que se realizaron en sprints seguindo a metodoloxía scrum mencionada no capítulo ENGADIRMENCIÓN. Neste capítulo trataremos de exlpicar con máis detalle cales foron. Na sección REFERENCIA podemos ver os requisitose historias que se explican no apartado REFERENCIA.

8.1 Análise de requisitos e historias do usuario

O sistema guardIAn deseñouse para cubrir necesidades clave de profesionais de ciberseguridade, aproveitando as capacidades dos LLM (modelos de linguaxe a gran escala) en tarefas do dominio. Estudos recentes amosan que os LLM son aplicables en múltiples ámbitos da ciberseguridade, incluíndo detección de vulnerabilidades, xeración de código seguro, intelixencia de ameazas, análise de logs en busca de anomalías e mesmo asistencia en ataques simulados. Neste proxecto, identificáronse os seguintes requisitos funcionais e non funcionais para guardIAn:

8.1.1 Requisitos funcionais

Os requisitos funcionais son especificacións que describen as funcións, servizos e comportamentos que un sistema debe realizar para cumprir cos obxectivos definidos polos usuarios ou interesados. Definen que debe facer o sistema fronte a diferentes entradas, situacións ou eventos.

- **Obtención e depuración dun corpus especializado:** Reunir, limpar e etiquetar logs de seguridade, entradas de foros técnicos especializados garantindo un conxunto de datos axeitado para adestrar o modelo. Os foros utilizados neste caso práctico foron Gentoo e Debian.

- **Adaptación do LLM ao dominio de ciberseguridade:** Aplicar fine-tuning ao LLM Llama 3.1 mediante LoRA para que entenda terminoloxía, tácticas e vulnerabilidades propias da información obtida dos foros.
- **Interface de usuario dual (web + API REST):** Proporcionar unha GUI que aínda que utiliza no backend Django, usa no frontend React para ter un mellor uso interactivo además dun servizo FastAPI para integracións automáticas con outras ferramentas de seguridade e garantir a robuste.
- **Xestión e administración de usuarios:** Permitir alta de contas, inicio de sesión, e creación dos diferentes chats para cada usuario.
- **Almacenamento e indexación intelixente dos datos cargados:** Gardar logs e fontes externas en ElasticSearch, xerando representacións vectoriais que posibiliten buscas semánticas e filtrado por metadatos mediante un índice.
- **Conexión entre o LLM e o almacén de coñecemento:** Construír unha pipeline RAG con LangChain que recupere fragmentos relevantes do índice e os incorpore ao razoamento do modelo en tempo real coa intención de mellorar substancialmente a inferencia.
- **Sistema de confiabilidade e referencias:** Incluír nas respostas indicadores de confianza e ligazóns ás evidencias ou documentos nos que se basea a explicación, reducindo o risco de alucinacións.

8.1.2 Requisitos non funcionais

Requisitos non funcionais son especificacións que definen criterios de calidade, restricións técnicas ou condicións operativas que o sistema debe cumprir, como o rendemento, a seguridade, a usabilidade, a escalabilidade, a compatibilidade, ou a dispoñibilidade. Neste proxecto identificáronse os seguintes requisitos:

- **Escalabilidade:** capacidade do sistema para soportar crecemento simultáneo de datos e usuarios sen perder rendemento, grazas a unha arquitectura de microservizos con despregue horizontal e indexación eficiente.
- **Eficacia e precisión:** capacidade de proporcionar respostas útiles e correctas que resolven o problema do usuario coa exactitude precisa para identifican eventos anómalos e evitar falsos positivos.
- **Extensibilidade:** ten que ser modular e flexible para permitir, con mínimos cambios no núcleo, tanto a substitución ou re-adestramento do LLM como a incorporación de novos

conectores e ferramentas, asegurando a súa adaptación continua ás ameazas futuras e protexendo a inversión.

- **Usabilidade:** debe ser doada e rápida de usar mediante unha interface clara e intuitiva con respostas ben estruturadas e con terminoloxía axeitada para a correcta comprensión.
- **Rendemento (velocidade de resposta):** as respostan teñen que xerarse en cuestión de segundos garantindo unha baixa latencia sen perder precisión nin calidade.

8.1.3 Historias de persoa usuaria

A partir dos requisitos funcionais definíronse varias historias de usuario que describen funcionalidades específicas dende a perspectiva dos usuarios finais (analistas blue team, pentesters red team ou administradores). Cada historia inclúe unha descrición e a estimación de esforzo en puntos. Considerando a planificación de 8 sprints cun máximo de 60 puntos por sprint (480 puntos en total), asignouse a cada historia unha valoración acorde á súa complexidade. Na táboa seguinte preséntanse as principais historias de usuario de guardIAn:

ID	Descrición da historia de usuario	Estimación (pts)
1	Como developer necesito familiarizarme cos fundamentos teóricos dos LLM aplicados á ciberseguridade, para poder deseñar unha solución sólida.	30
2	Como analyst e developer preciso avaliar e seleccionar as tecnoloxías (frameworks RAG, stack web e diferentes ferramentas) máis axeitadas para o proxecto.	20
3	Como product owner quero un corpus especializado (foros de Gentoo e Debian) que sirvan de base para o adestramento.	30
4	Como developer preciso revisar datasets e modelos de ciberseguridade, para identificar mellores prácticas e lagoas de investigación.	10
5	Como product owner quero finetunear Llama-3 con LoRA para adaptar o modelo ao dominio de ciberseguridade.	45
6	Como developer necesito profundizar en técnicas de fine-tuning e optimización (LoRA, cuantización, inferencia acelerada) para Llama-3.	20
7	Como developer quero expor o modelo a través dunha API REST construída con FastAPI, para que outras aplicacións poidan consumilo.	30
8	Como user desexo unha GUI web moderna (React sobre Django) que me permita conversar de forma intuitiva co chatbot.	25
9	Como product owner quero integrar guardIAn nunha plataforma profesional multi–equipo con xestión de contas e roles.	20
10	Como user quero conversar sobre código, obtendo análise contextual en linguaxe natural para comprender incidentes.	20
11	Como product owner preciso almacenar e indexar arquivos en ElasticSearch con embeddings para soportar buscas semánticas rápidas.	25
12	Como user quero recibir respostas confiables, reducindo alucinacións.	25
13	Como developer quero implementar un filtro de peticións ilícitas para garantir un uso ético da ferramenta.	20
14	Como developer e product owner necesito validar o sistema con profesionais de ciberseguridade e recoller feedback para mellora continua.	40
	Puntos totales	360

Tabla 8.1: Correspondencia entre historias da persoa usuaria e os puntos.

8.2 Realización dos sprints

A Continuación explicarase un por un como se fixo a realización de todos os sprints así como as datas e demais detalles relevantes atopados no proceso.

8.2.1 Sprint 1: Formación en LLMs + análise tecnolóxica

Duración: (08/01/25 ao 26/01/25)

O Sprint inicial do proxecto marca o punto de partida sobre o que se sustentará todo o traballo futuro. Nesta fase escóllense as historias 1, 2 e 3, centradas principalmente na formación e na análise preliminar, aspectos lóxicos nunha etapa inicial. Tamén se inclúe unha tarefa orientada ao comezo da construción do dataset preciso. Establécese unha base firme de coñecementos sobre LLMs e identificanse as ferramentas tecnolóxicas máis axeitadas, é clave para asegurar un desenvolvemento de calidade e uns resultados sólidos. Como parte deste proceso inicial, realizouse unha descomposición das historias en tarefas, identificando as seguintes:

Tarefas:

- **T1 – Estudo de fundamentos teóricos e claves.**
  - T1.1 – Estudo e asimilación de conceptos clave sobre LLMs.
  - T1.2 – Investigación sobre Intelixencia Artificial conversacional aplicada á ciberseguridade.
  - T1.3 – Estudo de técnicas de IA explicable para comprender as decisións dos LLM e análise do paradigma Retrieval Augmented Generation (RAG) e das súas vantaxes.
- **T2 - Análise de tecnoloxías e contorna actuais**
  - T2.1 – Busca, comparativa e estudo das librarías e frameworks adecuados.
  - T2.2 – Configuración dos diferentes contornos do proxecto.
- **T3 - Revisión do estado da arte en datos e modelos**
  - T3.1 – Busca de conxuntos de datos públicos relevantes para o fine-tuning no noso escenario de ciberseguridade.
  - T3.2 – Familiarización cos datasets seleccionados (tamaño, formato, calidade da información).
  - T3.3 – Revisión de modelos LLM destacados no dominio da ciberseguridade.

Sprint review:

Unha vez revisada a Sprint a velocity do proxecto viuse que é a adecuada sendo o resultado exitoso e alto. Esto será de utilidade por se no futuro se nos chega a presentar algún tipo de atraso.

8.2.2 Sprint 2: Elaboración do corpus de adestramento

Duración: (27/01/25 ao 12/02/25)

Unha vez completada a etapa inicial de análise conceptual e escolla das ferramentas de desenvolvemento, o traballo durante o Sprint 2 céntrase na creación dun corpus de datos especializado, necesario para a fase posterior de adestramento do modelo. O propósito central desta iteración é compilar e preparar información relevante no ámbito da ciberseguridade que permita afinar un modelo de linguaxe de base mediante técnicas de fine-tuning.

Como resultado do avance acadado na planificación anterior, contamos agora cunha maior dispoñibilidade temporal, o que permite adiantar tarefas previstas para fases futuras. Deste xeito, neste Sprint abórdase tanto a preparación e tratamento dos datos como o estudo das técnicas necesarias para o adestramento efectivo de LLMs.

As accións inclúen a limpeza, normalización e integración de fontes diversas nun corpus coherente, así como a revisión da súa calidade e consistencia. Ademais, realízase unha primeira aproximación aos métodos de adestramento axeitados, xunto coa elaboración de scripts de proba para axustar parámetros clave como o learning rate, número de epochs ou tamaño de batch.

Tarefas:

- **T4 – Preparación dun conxunto de datos especializado.**
  - T4.1 – Limpeza, preprocesado e normalización da información bruta recopilada.
  - T4.2 – Integración de distintas fontes de datos homoxeneizando os formatos nun corpus único.
  - T4.3 – Revisión da calidade e consistencia do conxunto de datos resultante, asegurando a integridade.
- **T5 – Aprendizaxe necesario para a o uso e entrenamento de LLMs.**
  - T5.1 – Identificación de métodos axeitados para o fine-tuning
  - T5.2 – Scripts de probas cos cos hiperparámetros e opcións de configuración(tamaño de batch, learning rate, epochs, etc.).

Sprint review:

8.2.3 Sprint 3: Fine-tuning do LLM

Duración: (13/02/25 ao 01/03/25)

O avance previsto neste Sprint destaca pola súa relevancia estratéxica, xa que o produto que se obterá – un modelo lingüístico especializado no ámbito da ciberseguridade – representa

o núcleo sobre o que se artellará o resto do sistema. Este compoñente, fundamental para a arquitectura global da solución, require unha dedicación notable tanto en termos de tempo como de recursos. Por este motivo, o Sprint céntrase nun único obxectivo principal, ao que se lle asigna a totalidade da capacidade dispoñible segundo a estimación feita para este ciclo de traballo.

A planificación detallada desta etapa inclúe a identificación de tarefas ben definidas, que permiten organizar o esforzo de forma progresiva e estruturada. En primeiro lugar, abórdase a adaptación ao contorno técnico onde se realizará o adestramento, o que implica a solicitude e estudo da infraestrutura dispoñible así como a preparación dunha imaxe de contedor axeitada (T6). A continuación, desenvólvese o proceso de afinado do modelo (T7), o cal abrangue desde a creación dun script específico baseado na técnica LoRA, ata a escolla da configuración óptima e a execución completa do adestramento, con seguimento continuo dos rexistros e métricas para garantir a calidade dos resultados.

Tarefas:

- **T6 - Familiarización co contorno do laboratorio para o adestramento**
  - T6.1 – Solicitude de uso e documentación da tecnoloxía empregada para o proceso de adestramento no laboratorio.
  - T6.2 – Preaparación dunha imaxe de Singularity.
- **T7 - Proceso de fine-tuning do LLM.**
  - T7.1 – Creación do script de adestramento empregando a técnica LoRA sobre o modelo base.
  - T7.2 – Selección da configuración máis axeitada.
  - T7.3 – Execución do adestramento completo e monitorización dos logs e métricas para verificar o rendemento do modelo afinado.

Sprint review:

8.2.4 Sprint 4: Interacción co LLM (1/2)

Duración: (02/03/25 ao 18/03/25)

Unha vez finalizado o Sprint anterior, neste novo ciclo de traballo céntrase o esforzo en avanzar cara a unha versión do sistema que permita unha interacción básica cun modelo de linguaxe a través dunha API REST. A prioridade inmediata recae na finalización da parte pendente dunha tarefa xa iniciada previamente, que se deixou incompleta, polo que se retoma agora como paso imprescindible. En paralelo, incorpórase unha nova funcionalidade que ten

como obxectivo comezar a expor o modelo mediante un servizo accesible externamente, e, co ánimo de recuperar parte do atraso acumulado e optimizar os recursos dispoñibles, engádese ademais unha tarefa relacionada coa integración do sistema nunha contorna compartida por varios equipos. O Sprint preséntase así especialmente ambicioso, apoiándose na marxe de avance gañada en fases anteriores. As tarefas específicas definidas para este período inclúen o deseño dos endpoints necesarios, a elaboración dos scripts para a súa implementación e a realización dunha proba de concepto que permita verificar a viabilidade da integración da API REST no funcionamento do chatbot.

Tarefas:

- **T8 – Implementación da API REST para o chatbot.**
  - T8.1 – Definición e deseño dos endpoints necesarios.
  - T8.2 – Creación dos scripts necesarios para a implementación.
  - T8.3 – Proba de concepto para integrar o sistema.

Sprint review:

8.2.5 Sprint 5: Interacción co LLM (2/2)

Duración: (19/03/25 ao 05/04/25)

Durante o Sprint, o equipo mantivo o foco en avanzar de xeito sostido e con expectativas máis axustadas á realidade do ritmo recente de desenvolvemento. A pesar de que no Sprint anterior se lograra habilitar unha API REST para a interacción co LLM, quedara pendente a recuperación da desviación acumulada. Neste novo ciclo optouse por unha planificación máis contida, engadindo un incremento moderado de carga (uns 10 puntos adicionais respecto á velocidade habitual). Así, centrouse o traballo nas historias 8 e 9, incorporando tamén a historia 10 como medida de compensación. As tarefas principais incluíron o desenvolvemento dunha interface web para comunicarse co LLM (React para o frontend e Django no backend), así como a integración de capacidades que permitan ao modelo tratar contido técnico do usuario como arquivos de logs e fragmentos de código, incorporando ademais o historial de conversa para manter a continuidade contextual nas interaccións.

Tarefas:

- **T9 – Comunicación co LLM mediante unha GUI web**
  - T9.1 – Deseño e implementación do frontend en React.
  - T9.2 – Deseño e implementación do backend en Django.
- **T10 – Capacidades de conversa con contido de logs e código.**

  - T10.1 – Integración do módulo para que o LLM poida cargar e analizar arquivos de logs de sistema e fragmentos de código do usuario durante a conversa.
  - T10.2 – Integración do historial do chat para obter un contexto completo da conversa

Sprint review:

8.2.6 Sprint 6: Incorporación de contexto e evidencias

Duración: (06/04/25 ao 22/04/25)
Tarefas:

- **T11 – Integración dun índice**
  - T11.1 – Escolla da tecnoloxía necesaria para conectalo (LangChain)
  - T11.2 – Implementación da lóxica de búsqueda e recuperación de fragmentos relevantes para engadir contexto ás consultas do usuario.
  - T11.3 – Parsing e indexación dos documentos e datos de ciberseguridade no índice, garantindo a representación axeitada do seu contido.
- **T12 – Sistema de confianza nas respostas do chatbot.**
  - T12.1 – Estudo das ferramentas dispoñíbeis para xerar métricas de confianza e extraer evidencias nas respostas do LLM (Chains).
  - T12.2 – Selección e implementación da solución máis axeitada.

Sprint review:

8.2.7 Sprint 7: Melloras finais e preparación da validación

Duración: (23/04/25 ao 09/05/25)
Tarefas:

-
  –

Sprint review:

8.2.8 Sprint 8: Validación con expertos e peche do proxecto

Duración: (10/05/25 ao 26/05/25)
Sprint review:

Capítulo 9

Resultado do desenvolvemento

Neste capítulo mostrarase o resultado final do proxecto completo e en funcionamento dende a perspectiva do usuario final da aplicación. Os casos de uso principais son

9.1 Aplicación funcionando

A aplicación actualmente podería pasar a un estado de producción sen a necesidade de maior supervisión. Calquera usuario que acceda á interfaz de registro pode crear a súa conta en segundos e autenticarse mediante a pantalla de login obtendo directamente acceso ao chat onde se pode utilizar o LLM adestrado e todos os servizos con normalidade. A Continuación móstranse varios pasos do seu uso normal:

ESTE CAPÍTULO ESTÁ SEN REMATAR PORQUE AÍNDA TEÑO QUE POR FOTOS DO FUNCIONAMIENTO DA WEB E QUEDAN DETALLES MENORES DO FRONT POR PULIR E EDITAR

9.2 Fontes de datos e software dispoñible OpenSource

Has fontes de datos curadas están publicas en ”Hugging Face” que é un coñecido repositorio, están accesas as fontes completas extraidas nos enlaces de:

Gentoo: PONER EL ENLACE AQUÍ

Debian: PONER EL ENLACE AQUÍ

O script utilizados para o adestramento está en:

LINK AO FICHEIRO DO ADESTRAMENTO

Capítulo 10

Conclusións, desarrollos futuros e relación co MUEI

Este será o capítulo final da memoria do proxecto, detallaremos contidos aprendidos, explicaremos a relación entre o proxecto e o mestrado e o impacto a nivel profesional.

10.1 Conclusións

A culminación deste proxecto supón moito máis ca a simple entrega dunha aplicación funcional; representa a validación práctica dun conxunto de coñecementos, metodoloxías e competencias adquiridos ao longo do Mestrado en Enxeñaría Informática. Durante os oito sprints, combináronse técnicas de desenvolvemento áxil, prácticas sólidas de enxeñaría de software e principios de ciberseguridade para construír un asistente baseado en LLM capaz de apoiar equipos blue e red nun contexto profesional real. O resultado é unha plataforma que, grazas á súa arquitectura modular (frontend React, backend Django/FastAPI, microservizos de IA e contedores Singularity), pode despregarse en producción sen supervisión adicional e escalar segundo a demanda da organización.

10.1.1 A relevancia dos obxectivos cumpridos

O percorrido desde a afinación do modelo (Sprint 3) ata a exposición dunha API REST (Sprints 4-5) e da GUI web (Sprint 5) puxo a proba a capacidade do equipo para integrar investigación, deseño e implementación nun calendario axustado. A integración dun índice semántico e dun mecanismo de confianza nas respostas (Sprint 6) engadiu a capa crítica de trazabilidade e explicación, imprescindible nun dominio tan sensible como a ciberseguridade. O sistema resultante permite: (1) consultar vulnerabilidades ou logs en linguaxe natural, (2) obter respostas xustificadas con evidencias e métricas de confianza, e (3) manter o contexto completo da conversación, incluídos ficheiros anexados polo usuario. Estes fitos demostran a

viabilidade do enfoque Retrieval-Augmented Generation (RAG) como soporte operativo para equipos SOC e pentesters.

10.1.2 Aprendizaxes técnicas e metodolóxicas

- **Aplicación de contornas reproducibles**
- **Fine-tuning eficiente con LoRA**
- **Arquitectura hexagonal e patróns de deseño**
- **Xestión áxil adaptada á realidade dun proxecto individual**

10.2 Relación das asignaturas do Máster co traballo

- **Calidad, seguridad y auditoría informática: Nivel de relación:** Muy directa porque si lo digo yo es asi

  O proxecto xira arredor da ciberseguridade (blue/red team, auditorías, pentesting). A asignatura achega os modelos, normativas e metodoloxías de seguridade e auditoría que o chatbot deberá coñecer e axudar a aplicar. Esta materia achegouche unha visión rigurosa dos modelos de calidade do proceso software, das normas ISO/IEC e das técnicas de auditoría profesional que se aplican cando se avalía a seguridade dun sistema. O chatbot ten que propoñer contramedidas; para facelo con solvencia necesita manexar o mesmo vocabulario técnico (confidencialidade, integridade, dispoñibilidade, trazabilidade, evidencias, plan de remediación) e os mesmos fluxos de traballo que se estudaron na asignatura. Así, o coñecemento adquirido sobre métricas de seguridade, plans de test, revisións de código e informes de auditoría convértese no motor conceptual que garante que cada resposta do asistente este aliñada cos estándares de referencia da industria e coas boas prácticas de goberno da ciberseguridade.

- **Informática como servicio: Nivel de relación:** Muy directa

  O proxecto depende de contedores, máquinas virtuais e servizos xestionados que se axustan ás tipoloxías SaaS vistas na asignatura. Alí analizáronse tamén requisitos de privacidade, localización do dato, cifrado e implicacións fundamentais cando se custodia información sensible de ciberincidentes. Ademais, a asignatura proporcionou criterios para escoller provedores, calcular custos operativos, deseñar arquitecturas de alta dispoñibilidade e implementar cadeas CI/CD sobre cloud, competencias que agora se aplican directamente para facer que o chatbot sexa escalable, resistente a fallos e cumpridor das políticas corporativas e normativas vixentes.

- **Recuperación de la información y Web semántica: Nivel de relación:** Muy directa

  O corazón técnico é unha pipeline de Retrieval-Augmented Generation, onde primeiro se localizan documentos pertinentes mediante embeddings densos e logo se xera a resposta cun LLM. Na asignatura estudáronse rastreadores, técnicas de crawling polite, compresión de índices invertidos, ranking baseado en contido e enlaces, learning-to-rank e avaliación de motores de busca, exactamente as pezas que agora se reempregan para construír os índices en ElasticSearch e mellorar a precisión das respostas. Ademais, a parte de Web Semántica achega RDF, ontoloxías e SPARQL, útiles se se queren etiquetar logs con metadatos ou vincular vulnerabilidades coñecidas como entidades Linked Data, enriquecendo así a base de coñecemento que o chatbot consultará.

- **Diseño de sistemas de información: Nivel de relación:** Muy directa

  O desenvolvemento dun chatbot distribuído composto por frontend React/Django, backend FastAPI e servizos de IA require unha arquitectura limpa, desacoplada e fácil de evolucionar. Patrones de deseño (factory, façade, adapter), capas hexagonais, métricas de acoplamento e cohesión, así como principios SOLID e DDD, foron tratados en profundidade nesta asignatura e constitúen o marco para organizar o código e os microservizos do proxecto. Tamén se abordou a accesibilidade e a calidade interna do software, polo que as recomendacións sobre interfaces inclusivas,cobertura e refactorización continua encaixan co obxectivo de entregar un produto open-source mantible, escalable e apto para contribuidores externos.

- **Dirección de proyectos: Nivel de relación:** Directa

  Adoptamos unha variante individual de Scrum con sprints de 2-3 semanas, puntos de historia e xestión de riscos; todo iso deriva dos coñecementos desta asignatura sobre marcos áxiles. Aprendiches a planificar alcance, tempo, custo, persoas e integración, así como a xestionar interesados, comunicación e calidade, elementos todos reflectidos na memoria: hitos, gráfico de Gantt, matriz de riscos con impacto/probabilidade, estimación de custos de persoal e de infraestrutura. Esa disciplina de xestión asegúrache que o proxecto se manteña viable, entregable no prazo e documentado segundo boas prácticas profesionais.

Apéndices

Apéndice A

Material adicional

Ejemplo de capítulo con formato de apéndice. El objetivo de este tipo de capítulo es incluir información adicional complementaria o bien elementos de consulta que por su tamaño no son adecuados dentro del texto principal, por ejemplo, un diagrama de clases particularmente grande.

A.1 Ejemplo de apartado en anejos

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nu-

llam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Etiam lobortis facilisis sem. Nullam nec mi et neque pharetra sollicitudin. Praesent imperdiet mi nec ante. Donec ullamcorper, felis non sodales commodo, lectus velit ultrices augue, a dignissim nibh lectus placerat pede. Vivamus nunc nunc, molestie ut, ultricies vel, semper in, velit. Ut porttitor. Praesent in sapien. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Duis fringilla tristique neque. Sed interdum libero ut metus. Pellentesque placerat. Nam rutrum augue a leo. Morbi sed elit sit amet ante lobortis sollicitudin. Praesent blandit blandit mauris. Praesent lectus tellus, aliquet aliquam, luctus a, egestas a, turpis. Mauris lacinia lorem sit amet ipsum. Nunc quis urna dictum turpis accumsan semper.

Bibliografía