

WORKSHOP PRESENCIAL PARA EQUIPOS TÉCNICOS

Desarrollo seguro con IA

Programar con Claude Code o Copilot sin abrir agujeros de seguridad en el código

Un día de trabajo para que desarrollo, arquitectura y seguridad se pongan de acuerdo en tres cosas: qué puede tocar el asistente, cuándo tiene que preguntar y qué pruebas bloquean un cambio inseguro antes del merge. El último bloque lo trabajamos sobre el código de la empresa.

5 a 8 h

presenciales, según el equipo

5

laboratorios prácticos

6

bloques de contenido

1

bloque sobre el código de la empresa

EL PROBLEMA DE PARTIDA

La IA ya está dentro del proceso de desarrollo y los controles se han quedado fuera

El equipo gana velocidad con Claude Code o GitHub Copilot. Con ellos han entrado cosas nuevas en el proceso: ficheros como AGENTS.md o CLAUDE.md que cambian el comportamiento del agente, Skills descargados de terceros, servidores MCP con credenciales y modos que ejecutan comandos sin pedir permiso.

El riesgo empieza antes del código que genera el asistente, en todo lo que lee, instala y puede ejecutar. Un fichero de configuración del IDE puede desactivar las confirmaciones, y un issue público puede acabar siendo una orden para el agente que corre en el pipeline. Y sigue después, cuando un pull request parece correcto porque compila.

En este workshop convertimos eso en decisiones concretas. Al terminar, el equipo sabe qué revisar antes de habilitar una capacidad, cómo limitarla y qué evidencia pedir antes de aceptar el resultado, y lo ha probado sobre su propio código.

Las cinco puertas que el equipo aprende a cerrar

01 Contexto

Qué instrucciones carga el asistente y quién puede cambiarlas.

02 Skills

Qué comportamiento se instala antes de leer una sola línea.

03 MCP

A qué sistemas, datos y credenciales llega el agente.

04 Autonomía

Qué ejecuta solo, qué pregunta y qué no debe ejecutar nunca.

05 Código generado

Qué pruebas y gates deciden si un cambio entra en main.

Cinco superficies y un sistema de control que el equipo prueba ese mismo día en su repositorio.

RESULTADO DEL WORKSHOP

Cinco decisiones que el equipo podrá tomar sin improvisar

Cada una se demuestra durante el día, primero en los laboratorios y luego en el repositorio del equipo. Reconocer el concepto en una diapositiva no cuenta.

01

Distinguir el modelo del orquestador

Saber si el riesgo viene de la respuesta del modelo o de la política que decide qué archivos carga, qué herramientas llama y con qué permisos.

Evidencia: mapa de confianza del asistente que usa el equipo.

02

Auditar contexto, Skills y MCP

Detectar instrucciones envenenadas, capacidades sobredimensionadas, descripciones de tools que intentan dirigir al agente y servidores remotos que aceptan tokens que no son suyos.

Evidencia: revisión de tres artefactos inseguros y decisión documentada.

03

Elegir el nivel de autonomía

Separar lectura, edición y ejecución. Aplicar ask-first, una lista de denegación y un sandbox de ficheros y red donde haya credenciales o impacto sobre el sistema.

Evidencia: el agente pide permiso ante env, instalaciones y cambios en tests.

04

Definir cuándo el código está terminado

Traducir el riesgo a una Definition of Done con unit, golden, smoke, end-to-end, SAST, secretos y dependencias, escrita en el fichero de instrucciones del proyecto.

Evidencia: definición reutilizable lista para incorporar al proyecto.

05

Revisar sin leer cada línea

Concentrar la revisión humana de un pull request generado en aserciones, YAML de CI, lockfiles, instrucciones del proyecto, configuración del asistente y cambios sensibles.

Evidencia: un cambio generado se bloquea con un test visto primero en rojo.

CASOS DE USO

Situaciones que cualquier equipo de desarrollo reconoce

Son decisiones que se toman a diario cuando el asistente ya trabaja sobre un repositorio, y casi todas tienen un incidente público detrás. Cada riesgo lleva su referencia en OWASP: LLM para aplicaciones con modelos (2025) y ASI para aplicaciones agénticas (2026).

CONTEXT POISONING • LLM01, ASI06**Un fichero del repo convence al agente para escribir SQL inseguro**

La instrucción parece una convención técnica más. El asistente la trata como autoridad y concatena parámetros en una consulta.

CONTROL QUE SE PRACTICA

Revisar las instrucciones como código, buscar caracteres ocultos y abrir un chat nuevo después de limpiar el contexto.

EXCESSIVE AGENCY • LLM06**Un MCP de solo lectura acaba con acceso a repositorios privados**

La configuración enseña un comando y un token. No enseña hasta dónde llega ese token ni lo que hace el servidor con él.

CONTROL QUE SE PRACTICA

Identidad propia del agente, mínimo privilegio, allowlist y leer el servidor antes de habilitarlo.

UNEXPECTED CODE EXECUTION · ASI05

El agente edita su propia configuración y deja de preguntar

Una instrucción escondida en el código que el agente lee le pide añadir una línea a la configuración del IDE o a mcp.json. A partir de ahí ejecuta comandos sin confirmación.

CONTROL QUE SE PRACTICA

Ficheros de configuración del asistente fuera del alcance de escritura del agente y revisados como código en cada pull request.

GOAL HIJACK · ASI01, ASI02

Un issue público da órdenes al agente del pipeline

Un workflow pasa el título de un issue al prompt de un agente con shell y acceso a los secretos del job. Quien escribe el issue decide qué se ejecuta.

CONTROL QUE SE PRACTICA

Texto de issues, PR y commits tratado como entrada no confiable, tools mínimas y el job que lee texto ajeno separado del que publica.

SUPPLY CHAIN · LLM03, ASI04

El asistente inventa una dependencia que existe por casualidad

El nombre suena bien. Alguien lo registró después y el agente lo añade al lockfile como si fuera una librería conocida.

CONTROL QUE SE PRACTICA

Todo nombre nuevo pasa por una persona que mira el registro, el mantenedor y el cambio de lockfile.

TEST TAMPERING

Los tests pasan porque el agente ha rebajado lo que comprobaban

El cambio queda en verde después de borrar una aserción, saltar un caso o refrescar un snapshot sin autorización.

CONTROL QUE SE PRACTICA

Red first, prohibido aflojar tests y revisión humana de aserciones, skips y goldens.

SENSITIVE INFORMATION DISCLOSURE · LLM02

Un secreto acaba en el contexto sin aparecer en el diff

Puede entrar al pegar un log, ejecutar env, leer un .env con una tool o guardar datos sensibles en el fichero de proyecto.

CONTROL QUE SE PRACTICA

Cuidado al pegar, env y .env denegados, redactar antes de pegar y rotar al momento si el secreto ya salió.

Lo que se enseña, bloque por bloque

El orden sigue el camino real de una petición: entra el contexto, el agente consigue capacidades, actúa en el portátil o en el pipeline y propone un cambio que tiene que pasar los controles.

01 Cómo funciona de verdad un asistente de desarrollo

El riesgo se ve distinto al separar el modelo del orquestador.

- Modelo, orquestador, tools y resultados de tools.
- Tokens, ventana de contexto y qué se pierde entre sesiones.
- Contexto estático frente a contexto dinámico.
- Fronteras de confianza dentro del IDE y del repositorio.
- Dónde cae cada riesgo en el OWASP Top 10 for LLM Applications 2025 y en el Top 10 for Agentic Applications 2026.

Caso real: OWASP clasifica EchoLeak como secuestro de objetivo (ASI01), el incidente de Amazon Q como mal uso de herramientas (ASI02) y el de Replit como agente descontrolado (ASI10).

02 Contexto envenenado y cadena de instrucciones

El texto que lee el agente puede acabar siendo política.

- AGENTS.md, CLAUDE.md e instrucciones de Copilot como código que ejecuta el modelo.
- Prompt injection indirecto desde repositorios, issues, documentos y resultados de tools.
- Caracteres invisibles, instrucciones escondidas y persistencia entre chats.
- Procedencia de la documentación interna, RAG y bases vectoriales.

Caso real: Rules File Backdoor, Pillar Security, 2025: Unicode invisible en ficheros de reglas de Cursor y Copilot. EchoLeak, CVE-2025-32711: instrucciones ocultas en un correo que Microsoft 365 Copilot procesa sin que el usuario haga clic.

03 Skills y MCP sin ampliar el perímetro a ciegas

Instalar una habilidad y conectar un servicio son dos decisiones distintas.

- Cuándo usar un Skill, cuándo usar MCP y cuándo basta un CLI.
- Tool poisoning dentro de nombres y descripciones.
- Scopes, tokens, identidad humana frente a identidad del agente.
- Autorización de un MCP remoto según la especificación: OAuth 2.1 con PKCE, tokens emitidos para ese servidor (RFC 8707 y RFC 9728) y prohibido reenviar tokens a otros servicios.
- Servidores locales por STDIO: toman las credenciales del entorno de quien los arranca.
- Pinning, revisión de terceros y retirada de capacidades al terminar.

Caso real: GitHub MCP, Invariant Labs, 2025: un issue público lleva al agente a copiar datos de repositorios privados en un PR público. MCPoison, CVE-2025-54136: Cursor ligaba la aprobación al nombre del servidor y no a lo que ejecutaba.

04 Autonomía, sandbox y secretos

El mismo prompt tiene más o menos riesgo según lo que el agente pueda ejecutar y alcanzar.

- Los cuatro niveles: lectura, preguntar, autoeditar y auto completo.
- Allowlist para lo cotidiano y deny list para operaciones con impacto.
- Sandbox de ficheros y de red para el agente y sus subprocessos; contenedor de desarrollo cuando la herramienta no lo trae.
- La configuración del asistente como superficie de ataque: un cambio en settings.json o mcp.json desactiva las preguntas.
- Credenciales de producción, shell, nube y cambios destructivos.
- Qué se puede pegar, qué se redacta y qué se rota.

Caso real: CVE-2025-53773: una inyección activa la autoaprobación de Copilot en VS Code y acaba en ejecución de código. Nx s1ngularity, 2025: paquetes maliciosos lanzan Claude Code, Gemini CLI y Amazon Q con los permisos desactivados. Replit, 2025: un agente borra la base de datos de producción durante una congelación de código.

05 Agentes en el pipeline

Un agente en CI hereda los permisos del job y lee texto que puede escribir cualquiera.

- Issues, pull requests y mensajes de commit como entrada no confiable aunque el workflow sea propio.
- Quién puede disparar el workflow, qué token recibe y qué tools tiene el agente.
- Caché compartida y publicación: separar el job que lee texto ajeno del que tiene los secretos de release.
- Tokens de CI con el alcance justo: el que construye no publica.

Caso real: PromptPwnd, Aikido, 2025: texto de issues inyectado en agentes de GitHub Actions; Google corrigió Gemini CLI en cuatro días. Cline, febrero de 2026: un título de issue acabó en una versión no autorizada publicada en npm durante unas ocho horas. Amazon Q, CVE-2025-8217: un token con demasiado alcance permitió meter código en una release.

06 DevSecOps y revisión de código que nadie leerá entero

Cambia quién escribe el código, pero los gates no se negocian.

- Definition of Done persistente en AGENTS.md o en el fichero de proyecto, y no en el chat.
- Unit, golden, smoke y end-to-end: qué demuestra cada capa.
- SAST, secret scanning, dependencias y slopsquatting.
- Qué revisa una persona en un pull request generado: aserciones, skips, CI, lockfile, instrucciones, configuración del asistente y zonas sensibles.
- Disciplina de git cuando varios agentes trabajan sobre el mismo checkout.

Caso real: Spracklen et al., USENIX Security 2025: al menos un 5,2 % de paquetes inventados en modelos comerciales y un 21,7 % en modelos abiertos, con 205.474 nombres distintos que cualquiera puede registrar.

Cinco laboratorios y cinco controles comprobados

Todas las prácticas siguen el mismo patrón: vemos el comportamiento inseguro una vez, aplicamos el control y demostramos que el resultado cambia. Los laboratorios usan un workspace preparado para romperse sin consecuencias. Después, el mismo patrón se repite sobre el repositorio de la empresa.

01

El proyecto que da malas instrucciones

El equipo carga un fichero envenenado, pide una búsqueda de productos y localiza por qué el agente propone SQL vulnerable.

Resultado: instrucciones revisadas, caracteres ocultos detectados y contexto limpio en una sesión nueva.

02

El Skill que nunca debería instalarse

Cada participante revisa un Skill que exige volcar el entorno y saltarse la confirmación humana.

Resultado: decisión de rechazo justificada por una línea concreta, y no por una corazonada.

03

Dos MCP que parecen iguales en la configuración

El equipo compara el servidor acotado con otro capaz de leer cualquier fichero del equipo.

Resultado: solo se habilita la allowlist y la petición de un secreto queda bloqueada.

04

El código generado frente al arnés

La IA implementa una función bajo una Definition of Done. Después se rompe a propósito para demostrar que el test de inyección falla.

Resultado: evidencia red first y condiciones claras para aceptar o rechazar el pull request.

05

El dial de autonomía bajo presión

El prompt intenta empujar al asistente a ejecutar env, instalar paquetes y tocar tests sin preguntar.

Resultado: modo ask-first y deny list comprobados en la herramienta real de cada participante.

La última parte la hacemos sobre el código de la empresa

Lo que hemos visto romperse en los laboratorios se busca después en un repositorio del equipo. Cada participante trabaja en su portátil y yo paso por los puestos para resolver dudas y desatascar.

01

Qué lee el asistente

Localizar AGENTS.md, CLAUDE.md e instrucciones de Copilot, decidir quién puede modificarlos y buscar caracteres invisibles con el comando que usamos por la mañana.

02

Qué tiene conectado

Listar los MCP, Skills y ficheros de configuración del asistente que afectan a ese repositorio, con el alcance real de cada token. Lo que nadie sepa justificar se desactiva ese mismo día.

03

Qué puede ejecutar

Configurar ask-first, la deny list y el sandbox en la herramienta que usa cada uno, y comprobar que el agente pregunta antes de leer el entorno o tocar un test.

04

Cuándo está terminado

Escribir la Definition of Done en el fichero de instrucciones del proyecto y dejar elegido el primer control que entrará por pull request, con su responsable.

Todo pasa en el portátil de cada participante y sobre su repositorio. El código no sale de la empresa, yo no lo copio ni me lo llevo, y no es una auditoría: no hay informe de hallazgos. Lo que se escribe en ese bloque se queda en el repositorio de la empresa.

Entre 5 y 8 horas, según el equipo

La agenda alterna explicación, decisión y práctica, y acaba sobre el código de la empresa.

Bloque 1

Trazar el sistema que ya usa el equipo

Modelo, orquestador, contexto y tools, con el mapa de riesgos de OWASP. Dibujamos las fronteras de confianza antes de hablar de controles.

Bloque 2

Romper la confianza en el contexto

Poisoning, AGENTS.md, instrucciones persistentes y laboratorio 1. El equipo comprueba cómo una regla del repositorio cambia el código.

Bloque 3

Decidir qué se conecta y con qué permisos

Skills, MCP y su autorización, identidad, autonomía, sandbox y secretos. Laboratorios 2, 3 y 5.

Bloque 4

Bloquear un cambio inseguro antes del merge

Agentes en el pipeline, Definition of Done, revisión de pull requests generados y laboratorio 4.

Bloque 5

El código de la empresa

Cada participante abre su repositorio y su asistente. Inventario de instrucciones y capacidades, permisos en la herramienta real y Definition of Done en el proyecto. Todos salen con un primer control elegido.

MATERIAL QUE SE QUEDA EN LA EMPRESA

El workshop se acaba y los controles se quedan

El material está hecho para reutilizarlo en el repositorio y en las conversaciones entre arquitectura, seguridad y desarrollo.

Definition of Done para cambios con IA

Plantilla lista para adaptar a AGENTS.md o al fichero de instrucciones del proyecto.

Tarjeta de control de una página

Contexto, Skills, MCP, autonomía, secretos, código y supply chain, con su referencia OWASP.

Guía de higiene de contexto

Qué redactar antes de pegar y qué hacer cuando un secreto ya se compartió.

Lista de revisión para agentes en CI

Entradas no confiables, permisos del token, tools del agente, caché y separación del job de publicación.

Pipeline de referencia

Ejemplo para Azure DevOps con pytest real y los siguientes gates identificados.

Workspace con cinco laboratorios

Enunciados, fixtures seguros e inseguros y tests para repetirlos.

Deny list inicial

Operaciones de git, shell, cloud y secretos que requieren una persona.

Matriz de autonomía

Equivalencia de modos, aprobaciones y sandbox en Copilot, Claude Code y Gemini CLI.

Disciplina de git multiagente

Rutas explícitas, estado antes de commit y propiedad de cada fichero.

LO QUE SE SUMA AL CURSO

Cuatro cosas que se quedan con el equipo, sin coste añadido

Son lo que un equipo acaba pidiendo después del curso.

+1

Paquete de evidencia de cumplimiento

Lista de asistencia firmada, certificado individual con fecha, horas y contenidos, temario mapeado a las normas que apliquen a la empresa y una carta de una página para entregar al auditor.

+2

Encuesta previa de 10 minutos

Antes del día, el equipo responde una encuesta corta sobre su stack y dónde se atasca. Con ella adapto los laboratorios a lo que el equipo usa de verdad.

+3

Configuraciones en una rama del repositorio de la empresa

Durante el bloque final dejen en una rama las configuraciones que hemos probado: instrucciones y permisos del asistente, reglas de MCP y comprobaciones del pipeline. Incluye una plantilla técnica de política de uso de IA y MCP; es técnica, no asesoramiento jurídico. El equipo las revisa y las integra como cualquier otro cambio.

+4

Laboratorios con soluciones

El repositorio de laboratorios con sus soluciones y una licencia de uso interno de 12 meses, para que quien se incorpore al equipo no dependa de repetir el día.

CUMPLIMIENTO

Qué pide cada norma y qué entrego como evidencia

El curso no cumple ninguna norma por sí solo. Sirve como evidencia documentada para el control de formación que el auditor ya pide, si la norma aplica a la empresa.

NORMA	QUÉ PIDE	QUÉ DEJO COMO EVIDENCIA
PCI DSS 4.0 · 6.2.2	Formación al menos cada 12 meses para quien desarrolla software a medida: diseño seguro, codificación segura y herramientas de testing, documentada.	Temario mapeado a esos puntos, asistencia firmada y fecha.
ISO 27001:2022 · A.6.3	Concienciación, educación y formación en seguridad adecuadas a la función de cada persona.	Certificado individual con fecha, horas y contenidos, y registro por persona.
ISO 27001:2022 · A.8.25 y A.8.28	Ciclo de vida de desarrollo seguro y codificación segura, incluida la formación de quien desarrolla.	Temario mapeado y las comprobaciones de pipeline que quedan en la rama de la empresa.
SOC 2 · CC1.4 y CC2.2	Programa de formación documentado, con registros de finalización por persona y profundidad según el rol.	Lista con nombre, fecha, contenido y rol.
DORA · art. 13(6)	Programas de formación en resiliencia operativa digital para el personal, proporcionados a su función (banca y seguros).	Registro de asistencia y temario del curso.
AI Act · art. 4	Medidas para apoyar la alfabetización en IA del personal, adaptadas a su función. Es una obligación de medios y basta un registro interno.	Registro interno de la formación, con contenido sobre riesgos y uso de IA según la función.

Para organizaciones que han pasado de probar la IA a depender de ella

Encaja especialmente con

- Responsables de arquitectura de seguridad que necesitan convertir los riesgos de la IA en controles que desarrollo pueda aplicar.
- AppSec, DevSecOps y Platform Engineering, responsables de los gates del pull request, de los pipelines y de las conexiones con herramientas.
- Engineering managers y tech leads que ya ven código generado en repositorios de producción.
- Security champions y desarrolladores senior que van a extender estas prácticas al resto del equipo.
- Equipos que usan Claude Code o GitHub Copilot y todavía deciden los permisos caso por caso.

No es la formación adecuada si

- El equipo aún no ha usado un asistente sobre un repositorio real.
- La empresa busca una introducción a qué es un LLM o a cómo escribir prompts.
- Lo que hace falta es una política corporativa, gobierno de modelos o una formación para dirección.
- Lo que se espera es una auditoría del código con informe de vulnerabilidades. En el último bloque el equipo trabaja sobre su repositorio, pero la revisión la hace el propio equipo.

Seguridad y desarrollo contados por alguien que ha trabajado en los dos lados



Soy Daniel Alfocea y llevo veinte años en desarrollo y seguridad. En Banco Santander fui Principal Engineer del departamento de IA durante la puesta en marcha del área. En BBVA fui Tech Lead de ciberseguridad. En 42Crunch, en Reino Unido, fui responsable de uno de sus productos de seguridad de APIs.

He presentado investigación en RSA Conference y en OWASP AppSec EU, y he sido ponente habitual de RootedCON. El Instituto Nacional de Ciberseguridad (INCIBE) me ha llamado varias veces como juez de sus concursos. Fundé Navaja Negra y el capítulo de OWASP Madrid. Soy autor de herramientas de seguridad muy usadas, como GoLismero, Plecost y Dockerscan, y de LAPIs, una investigación publicada en arXiv sobre cómo describirle una API a un modelo con menos tokens. Con Alfonso Muñoz escribí *MCP Seguro. Guía práctica para desarrolladores, arquitectos software y tech leads*.

Este workshop junta dos cosas que casi siempre se enseñan por separado: cómo trabaja el asistente dentro del repositorio y cómo se construyen controles que un equipo de desarrollo pueda cumplir. Y acaba con esos controles aplicados sobre el código de la empresa.

GARANTÍAS

Dos compromisos por escrito y un límite

Si las configuraciones no funcionan, las arreglo

Si las configuraciones que dejo en el repositorio de la empresa no funcionan, hago una sesión remota gratuita para arreglarlas.

Una hora gratis si el equipo se atasca en 30 días

Si a los 30 días el equipo se ha atascado aplicando algo del curso, le dedico una hora en remoto sin coste.

Esto no es una auditoría ni una promesa de resultados: no garantizo porcentajes de mejora ni que no haya incidentes.

Preguntas que conviene resolver ahora

¿Cuánto cuesta?

Depende de la duración, entre cinco y ocho horas, y del tamaño del equipo. Tras la primera conversación mando una propuesta cerrada.

¿Es una formación para seguridad o para desarrollo?

Para las dos, siempre que compartan decisiones sobre el SDLC. Arquitectura y AppSec ponen los límites, y desarrollo tiene que convertirlos en instrucciones, tests y gates que pueda cumplir todos los días. Si solo viene una de las dos partes, el resultado pierde fuerza.

¿Hace falta instalar Claude Code y Copilot?

No. Cada participante necesita una de las dos herramientas, ya iniciada y capaz de trabajar contra una carpeta local. Las ideas son las mismas y solo cambian los nombres de los ficheros y de los modos.

¿Se adapta a nuestro lenguaje o stack?

Los laboratorios usan Python y el ejemplo de pipeline usa Azure DevOps. Los controles se aplican igual a otros lenguajes y sistemas de CI, y en el último bloque cada participante los aplica sobre el stack real del equipo.

¿Podemos usar nuestro repositorio durante el día?

Normalmente sí, en el último bloque. Los laboratorios usan un workspace preparado para romperse sin consecuencias, así todo el grupo ve el mismo fallo. Después, cada participante abre un repositorio de su equipo en su portátil y aplica lo aprendido. Si la empresa no puede abrir su código ese día, trabajamos sobre un repositorio de práctica parecido a su stack. El código no sale de la empresa.

¿Qué tiene que estar resuelto antes de ese bloque?

Que cada participante pueda abrir el repositorio con su asistente según la política interna de la empresa. Si un repositorio no puede pasar por esa herramienta, se elige otro del equipo que sí pueda. Lo confirmamos en la primera conversación.

¿Se puede bonificar por FUNDAE?

Al pedir fecha paso la información que necesita la gestoría o el departamento de formación.

¿Se puede impartir en remoto?

Sí, aunque en presencial funciona mejor para revisar configuraciones y desatascar prácticas. El alcance es el mismo.

¿Garantiza que no tendremos incidentes con IA?

No. Garantizo los cinco laboratorios, el bloque sobre el código de la empresa y la entrega del material. Que baje el riesgo depende de que la organización aplique y mantenga los controles después.

¿Y si necesitamos seguimiento?

Existe un programa de 6 semanas de esta misma temática, con medición antes y después y revisión de los PR del equipo. El detalle está en [el programa](#).

FORMATO

Un equipo, una sala y un sistema común para decidir

Incluye la preparación, entre cinco y ocho horas presenciales según el equipo, el bloque de trabajo sobre el repositorio de la empresa y todo el material. Con grupos pequeños puedo revisar los laboratorios mientras se hacen y pasar por cada puesto en el último bloque. Con demasiada gente, la práctica se queda en demostración.

- Participantes del mismo equipo o de áreas que trabajan juntas.
- Lo imparto en español o en inglés.
- Sala, pantalla y wifi a cargo de la empresa.
- Cada participante trae su portátil, con su asistente y acceso a un repositorio del equipo.
- Desplazamiento fuera de Madrid acordado antes de cerrar la fecha.
- Incluye el paquete de evidencia, las configuraciones en la rama de la empresa y los dos compromisos de arriba.
- Está pensado como un día de trabajo para un equipo que va a aplicar lo aprendido. No es una auditoría, ni un certificado de cumplimiento, ni una charla para escuchar.

Escríbeme a

yo@danielalfocea.com

En la primera conversación vemos si el workshop encaja, cuántas personas van a participar y qué fecha le viene bien a la empresa.

Bonificable a través de FUNDAE.

CORREO

yo@danielalfocea.com

PROGRAMA EN LA WEB

danielalfocea.com/formacion/workshop-desarrollo-seguro-con-ia