Daniel Alfocea

WORKSHOP PARA EQUIPOS DE DESARROLLO, PRESENCIAL U ONLINE

Productividad real con IA

Saca más trabajo de Claude Code o Copilot, gasta menos tokens y repasa menos

Un día con Claude Code y GitHub Copilot para que tu equipo saque más trabajo, gaste menos tokens, se fíe de lo que sube a producción y trabaje igual en todos los repositorios.

5 a 8 h	6	2	8
presencial u online, según el equipo	laboratorios con medición antes y después	herramientas: Claude Code y GitHub Copilot	bloques de temario avanzado

danielalfocea.com/formacion/workshop-productividad-con-ia

yo@danielalfocea.com

EL PROBLEMA QUE YA TENÉIS

Todo el equipo usa IA, cada uno a su manera, y la factura se nota

Uno usa el modelo más caro para renombrar una variable. Otro lleva una sesión abierta desde el lunes que reenvía todo su historial en cada mensaje. Otro deja al agente toda la tarde en bucle, intentando arreglar un test que no puede pasar. Y los pull requests llegan en verde porque el mismo agente que escribió el código decidió que estaba bien.

La herramienta no tiene la culpa. Claude Code y Copilot traen límites de turnos, topes de gasto, hooks, subagentes con su propio modelo y revisores en contexto limpio. Casi nadie lo configura, porque está desperdigado entre la documentación, los changelogs y los issues del repositorio.

Este día va de eso: de las piezas que deciden si el asistente te ahorra trabajo o te lo da. **Cada una se mide en tus propias sesiones, antes y después**, así que el ahorro no te lo prometo yo, lo ves tú.

Lo que el equipo se lleva del día

01 **Más trabajo terminado**
El modelo caro solo donde hay que pensar, y agentes que trabajan en paralelo sin pisarse.

02 **Menos tokens**
Una caché que no se rompe, contexto que se carga cuando hace falta y salidas que no te llenan la ventana.

03 **Sesiones que terminan**
Límite de turnos y de gasto, y una condición de parada que funciona aunque nadie esté mirando.

04 **Confianza en producción**
La IA prueba lo que genera, y otro agente intenta romperlo antes de que llegue al merge.

05 **Un método común**
La misma configuración y las mismas reglas en todos los equipos.

Todo lo enseño en Claude Code y en GitHub Copilot. Al final del día, cada uno se lo lleva configurado en su propio repositorio.

RESULTADO DEL WORKSHOP

Ocho cosas que tu equipo sabrá hacer el lunes

Cada una se practica durante el día con medición, y en el último bloque queda configurada en el repositorio de cada participante.

01

Elegir modelo con datos y pagar el caro solo donde piensa

Comparar modelos con tus propias tareas en vez de fiarte del ranking. Planificar con el modelo grande y ejecutar con uno más barato, subagentes que exploran con Haiku y un esfuerzo de razonamiento distinto por tarea.

Evidencia: una tabla con tres modelos sobre la misma tarea, con tests que pasan, coste y tiempo de cada uno.

02

Gastar menos tokens sin perder calidad

Saber qué rompe la caché, por qué rebobinar sale más barato que compactar y cómo filtrar la salida de las herramientas antes de que llegue al modelo.

Evidencia: tokens de entrada, salida y caché medidos en la sesión de cada participante.

03

Dar contexto solo cuando hace falta, sin RAG

Un fichero de instrucciones corto que hace de índice, reglas que solo se cargan cuando tocas su ruta y documentación que el agente encuentra por su cuenta.

Evidencia: una regla centinela que demuestra qué ficheros lee de verdad cada herramienta.

04

Dejar currando al agente y que termine

Límite de turnos y de presupuesto por sesión, una condición de parada que evalúa otro modelo y el bucle Ralph: el mismo encargo en vueltas con contexto limpio, el progreso guardado en el repositorio y un tope de iteraciones.

Evidencia: una tarea autónoma que se detiene sola al cumplir su objetivo o al agotar su límite.

05

Cortar los bucles infinitos

Reconocer el bucle antes de que te queme la tarde, y dejar configurado lo que lo corta aunque nadie esté mirando.

Evidencia: un bucle provocado a propósito y cortado por la configuración, sin que nadie intervenga.

06

Fiarse de lo que va a producción

Que la IA pruebe su propio código: un test que falla primero, tests protegidos contra ediciones del agente y una suite que se ejecuta antes de dar nada por terminado.

Evidencia: un agente que intenta hacer trampa con los tests y una configuración que se lo impide.

07

Que los agentes se vigilen entre sí

Un revisor en contexto limpio, validación cruzada con modelos distintos, un juez con sesgos controlados y un agente adversario que solo sirve para romper cosas.

Evidencia: un panel que encuentra fallos sembrados que el autor dio por buenos.

08

Trabajar igual en todo el equipo

Configuración gestionada para toda la organización, un catálogo propio de skills y agentes, y una sola carpeta que leen Claude Code y Copilot.

Evidencia: la misma configuración funcionando en las dos herramientas del equipo.

CASOS DE USO

Cinco fallos documentados que te cuestan dinero

Salen en los repositorios y las guías de las propias herramientas, y aparecen en cuanto el agente trabaja sobre un proyecto de verdad.

BUCLE SIN FIN

El agente reintenta lo mismo durante horas

Hay un issue público de Claude Code con más de 300 reintentos seguidos, 27 millones de tokens y 4,6 horas de ejecución. Nadie lo paró porque nadie estaba mirando.

CONTROL QUE SE PRACTICA
Límite de turnos y de gasto por sesión, un hook de parada con guarda y un tope de continuaciones.

MODELO CARO PARA TODO

El modelo más potente busca ficheros y renombra variables

Explorar un repositorio no necesita al modelo que diseña la arquitectura, pero lo usa porque es el que está seleccionado.

CONTROL QUE SE PRACTICA
Modelo por ronda: el grande planifica y revisa, uno barato explora y ejecuta, cada subagente con su modelo fijado.

CACHÉ ROTA

La misma sesión cuesta el doble y nadie sabe por qué

Cambiar de modelo a mitad de conversación, activar el modo rápido o compactar invalida la caché, y todo el historial se vuelve a pagar a precio completo.

CONTROL QUE SE PRACTICA
Saber qué rompe la caché, medirla en la barra de estado y rebobinar en vez de compactar cuando se puede.

FALSO VERDE

Los tests pasan porque el agente los ha cambiado

Ante un test que no consigue pasar, el agente relaja la aserción, lo marca como saltado o hace que la función devuelva justo lo que el test espera.

CONTROL QUE SE PRACTICA
Tests protegidos con un hook que bloquea su edición, rojo antes que verde y un revisor que no ha visto la conversación del autor.

CADA UNO CON SU CONFIGURACIÓN

Diez desarrolladores, diez formas de trabajar con agentes

Cada uno tiene sus permisos, sus prompts y su modelo por defecto. Nadie sabe qué funciona ni cuánto cuesta.

CONTROL QUE SE PRACTICA

Configuración gestionada, catálogo común de skills y agentes y una carpeta compartida por las dos herramientas.

PROGRAMA COMPLETO

Lo que se enseña, bloque por bloque

Cada bloque lo explico en Claude Code y en GitHub Copilot. Lo básico lo repaso rápido, porque ya usáis las herramientas a diario, y el tiempo se va en lo que casi nadie configura. Con cinco horas, los bloques 6 a 8 se condensan.

01 Diagnóstico: medir antes de optimizar

Lo que no ves no lo mejoras. Primero medimos cada sesión.

- Qué ocupa la ventana y cuánto cuesta la sesión: /context, /usage y una barra de estado con coste y caché en tiempo real.
- Coste por ejecución en modo no interactivo, leyendo el JSON que devuelve claude -p.
- Cuánto dura la caché según cómo uses Claude Code, y por qué eso cambia cuándo conviene hacer una pausa.
- En Copilot: el sistema de créditos vigente, qué consume y qué es gratis, como el autocompletado.

> **Caso real:** La documentación de costes de Claude Code da una media de unos 13 dólares por desarrollador y día activo. El primer ejercicio es medir dónde está tu equipo.

02 Qué modelo usar y cómo elegirlo

El primero del ranking no tiene por qué ser el mejor para tu repositorio. El modelo se elige midiendo.

- Qué modelos hay en cada herramienta: Claude Code trae los de Anthropic; Copilot, los de Anthropic, OpenAI y Google, cada uno con su multiplicador.
- Los criterios que cuentan: calidad en vuestras tareas, coste de la tarea entera y no el precio por token, ventana de contexto, latencia y esfuerzo de razonamiento.
- Una evaluación propia: tareas reales del backlog, dos o tres modelos y una tabla con tests que pasan, coste y tiempo. Cada modelo nuevo pasa la misma batería antes de cambiar el de por defecto.
- Planificar con el grande y ejecutar con otro, subagentes con su modelo fijado y el explorador integrado de Claude Code trabajando con Haiku.
- En Copilot: agentes con lista de modelos de reserva y el modelo auxiliar que nadie configura.

> **Caso real:** Un modelo barato que necesita el triple de turnos sale más caro que uno caro que acierta a la primera. Y el modo rápido no ahorra: cuesta más por token y se usa cuando importa la latencia.

03 Menos tokens de entrada y de salida

Casi todo el gasto está en lo que se reenvía, no en lo que escribes.

- Qué invalida la caché: cambiar de modelo o de esfuerzo, el modo rápido y compactar.
- Rebobinar frente a compactar, y cuándo empezar de cero con un resumen escrito.
- Hooks que filtran la salida de los comandos antes de que llegue al modelo.
- Herramientas MCP bajo demanda, y en Copilot el límite de herramientas por petición y cómo reducir la lista.

> **Caso real:** Un hook que recorta la salida de los tests a las líneas que fallan deja la misma información con una fracción de tokens.

04 Contexto solo cuando hace falta, sin RAG

El agente no tiene que leerlo todo, tiene que saber dónde está cada cosa.

- Un fichero de instrucciones de menos de doscientas líneas que hace de índice, y reglas que se cargan solo al tocar su ruta.
- Qué se carga de verdad: los imports no son perezosos y los comentarios HTML no llegan al modelo.
- Documentación organizada por niveles que el agente recorre solo, sin base de datos vectorial.
- En Copilot: instrucciones por ruta, y qué ficheros de Claude Code lee también, comprobado con una regla centinela.

> **Caso real:** Si hay CLAUDE.md, Claude Code no lee AGENTS.md. Una línea centinela lo demuestra en diez segundos.

05 Ejecución finita: sesiones que terminan

Un agente sin límite sale caro, y de autónomo tiene poco.

- Máximo de turnos y de presupuesto por ejecución, con un código de salida distinto para cada corte.
- El bucle Ralph: el mismo encargo en vueltas sucesivas, cada una con contexto limpio, el progreso escrito en ficheros del repositorio, una promesa de finalización y un máximo de iteraciones.
- Objetivos con condición de parada evaluada por otro modelo, y hooks de parada con su guarda contra el bucle.
- En Copilot: el límite de peticiones del modo agente, el piloto automático con tope de continuaciones y el límite del agente en la nube.

> **Caso real:** El bucle Ralph funciona porque cada vuelta empieza sin el historial de la anterior. Sin tope de iteraciones ni criterio de terminado, es la forma más rápida de quemar tokens.

06 Orquestar grupos de agentes

Más agentes no es mejor. Lo que funciona son agentes distintos con encargos que no se pisan.

- Worktrees por agente, trabajo en lote sobre muchos ficheros y una sesión por pull request.
- Panel de expertos con roles disjuntos y validación cruzada entre modelos distintos.
- Juez con control de sesgos de posición, longitud y autopreferencia, y agente adversario con un único encargo: romper.
- Cuándo no montar un sistema multiagente, y en Copilot los traspasos, la flota de agentes de la CLI y el agente en la nube.

> **Caso real:** Anthropic midió su sistema multiagente de investigación: un 90 % más de rendimiento a cambio de unas quince veces más tokens. Solo compensa en ciertas tareas.

07 Que la IA pruebe su propio código

Un cambio está terminado cuando lo dice una comprobación, no cuando lo dice el agente.

- Test que falla primero, en la línea esperada, y un hook que impide al agente editar los tests.
- Una puerta de parada que ejecuta la suite y no deja cerrar la tarea con nada en rojo.
- Revisor en contexto limpio, revisión de seguridad automática y mutation testing para saber si los tests prueban algo.
- En Copilot: el agente en la nube valida con CodeQL, y la aprobación automática del terminal no es una frontera de seguridad.

> **Caso real:** Un modelo que se autocorrige sin información externa no mejora su razonamiento. Por eso el revisor no puede compartir contexto con el autor.

08 Una metodología común para todos los equipos

Lo que funciona en un repositorio tiene que funcionar en todos, sin depender de quién lo configuró.

- Configuración gestionada para la organización: modelos permitidos, modos peligrosos desactivados y permisos comunes.
- Catálogo propio de skills, agentes y hooks que se instala igual en cada repositorio.
- Una sola carpeta de configuración que leen Claude Code y Copilot, y telemetría por equipo.
- En Copilot: qué instrucciones de organización no llegan al IDE y dónde no aplican las exclusiones de contenido.

> **Caso real:** Las exclusiones de contenido de Copilot no se aplican en el modo agente ni en la CLI. Si alguien cuenta con ellas, mejor enterarse hoy.

PRÁCTICA DELIBERADA

Seis laboratorios, cada uno medido antes y después

Cada práctica se hace sobre un repositorio de ejercicios que llevo preparado, en Claude Code y en Copilot. Cada uno apunta sus números antes y después. El ahorro se mide, yo no lo prometo.

01

Elegir modelo midiendo

La misma tarea se resuelve con tres modelos distintos y después con planificación, exploración, ejecución y revisión repartidas entre modelos.

Resultado: una tabla con tests que pasan, coste y tiempo de cada opción, que el equipo reutiliza como batería para el próximo modelo.

02

La caché y la salida

Se rompe la caché a propósito de tres formas distintas y se mide cada una. Después se añade un hook que filtra la salida de los comandos.

Resultado: tokens de caché, entrada y salida medidos, y un hook que el equipo puede copiar.

03

Contexto a dieta

Un fichero de instrucciones largo y contradictorio se convierte en un índice con reglas por ruta. Una regla centinela demuestra qué lee cada herramienta.

Resultado: menos contexto cargado y una regla que el agente cumple y antes ignoraba.

04

Hacerlo finito

Una tarea autónoma se lanza sin límites y entra en bucle. Se repite como bucle Ralph, con el progreso en un fichero, tope de iteraciones, límite de gasto y una condición de parada evaluada por otro modelo.

Resultado: una ejecución que termina sola, con el coste de las dos versiones comparado.

05

Pillar al tramposo

El repositorio trae fallos sembrados. Se pide al agente que haga pasar los tests y se observa cómo intenta conseguirlo. Después se protegen los tests y se pone una puerta de parada.

Resultado: los fallos sembrados aparecen y el agente ya no puede dar la tarea por terminada en rojo.

06

Panel y juez

Tres agentes con roles distintos y modelos distintos revisan el mismo cambio. Un juez decide, y un agente adversario intenta romper lo que todos aprobaron.

Resultado: fallos encontrados que el autor dio por buenos, y una plantilla de panel lista para usar.

CON VUESTRO CÓDIGO

El último bloque, en vuestro repositorio

Al final del día cada participante abre su portátil y configura su propio repositorio con lo practicado. Yo voy pasando, resuelvo dudas y reviso decisiones cuando alguien me lo pide.

01

Medir el punto de partida

Contexto ocupado, coste de una sesión normal y uso de caché, con la barra de estado que hemos montado durante el día.

02

Instrucciones, modelos y límites

Fichero de instrucciones como índice, subagentes con su modelo fijado y límites de turnos y de gasto para lo que se lanza en autónomo.

03

Una tarea real con su puerta

Una tarea pequeña del backlog propio: test que falla primero, tests protegidos, suite antes de cerrar y un revisor en contexto limpio.

04

Medir otra vez y dejarlo por escrito

Los mismos números que al principio, apuntados junto a una lista corta de lo que el equipo quiere estandarizar.

> Se trabaja en el portátil de cada participante y con sus herramientas de siempre. El código no sale de la empresa, no me lo llevo ni lo copio a ningún sitio. Tampoco es una auditoría: cada uno decide qué enseña.

ARQUITECTURA DEL DÍA

Entre 5 y 8 horas, según el equipo

La duración se ajusta a tu equipo y a lo que más le duele. El orden no cambia: medir, ahorrar, acotar, verificar y aplicarlo al repositorio propio.

Bloque 1	**Medir y elegir modelo** Diagnóstico de la sesión de cada participante, cómo elegir modelo y qué modelo y esfuerzo usar en cada ronda. Laboratorio 1.
Bloque 2	**Menos tokens y contexto solo cuando hace falta** Caché, salida filtrada, instrucciones como índice y reglas por ruta. Laboratorios 2 y 3.
Bloque 3	**Sesiones que terminan** Límites de turnos y de gasto, bucle Ralph, condiciones de parada y cómo cortar un bucle. Laboratorio 4.
Bloque 4	**Verificar y orquestar** Que la IA pruebe su código, tests protegidos, panel de agentes, juez y adversario. Laboratorios 5 y 6.
Bloque 5	**Vuestro repositorio** Cada participante mide, configura y vuelve a medir su propio proyecto. Cierre con lo que el equipo quiere estandarizar.

MATERIAL QUE SE QUEDA EN LA EMPRESA

Lo que queda en vuestro repositorio

Todo el material está hecho para copiarlo al repositorio y mantenerlo ahí. Si acaba en una carpeta de formación sin abrir, he fallado.

Barra de estado con coste
Coste de la sesión, contexto ocupado y caché a la vista en cada momento.

Repositorio con seis laboratorios
Enunciados, ficheros de partida, fallos sembrados y tests para repetir cada práctica.

Subagentes listos para usar
Explorador barato, revisor en contexto limpio, adversario y juez, con modelo y límite de turnos fijados.

Hooks de ejemplo
Filtro de salida, protección de tests y puerta de parada, para Claude Code y para Copilot.

Plantillas de instrucciones
Fichero índice, reglas por ruta y la regla centinela, en una carpeta que leen las dos herramientas.

Recetas de ejecución finita
Bucle Ralph con su fichero de progreso, límites de turnos y de gasto, condiciones de parada y topes del modo autónomo en las dos herramientas.

Plantilla de encargo a un agente
Qué hacer, qué no hacer, límite de turnos y formato de informe.

Batería de evaluación de modelos
Las tareas y la tabla del laboratorio 1, para probar cada modelo nuevo antes de cambiar el de por defecto.

Configuración base para la organización
Modelos permitidos, permisos comunes y modos peligrosos desactivados, para empezar a estandarizar.

LO QUE SE SUMA AL CURSO

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

Van incluidas en el workshop. Es lo que un equipo acaba pidiendo después.

+1

Encuesta previa de 10 minutos

Antes del día, el equipo responde una encuesta corta sobre su stack, sus herramientas y dónde se le va el tiempo. Con ella adapto los laboratorios a lo que usáis de verdad.

+2

Los números del equipo

Al empezar, cada participante apunta tres números de una sesión suya normal: tokens, porcentaje de caché y coste (en Copilot, peticiones premium). Los vuelve a apuntar después de cada laboratorio y al final del día, en una plantilla compartida que llevo preparada. Esa semana os mando una página con los números del equipo: dónde se iba el gasto y qué ha cambiado.

+3

Configuraciones en una rama de vuestro repositorio

Durante el bloque final dejo en una rama las configuraciones que hemos probado: instrucciones, subagentes, hooks y límites. Las revisáis e integráis 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 llegue al equipo no tenga que esperar a que repitamos el día.

A QUIÉN VA DIRIGIDO

Para equipos que ya programan con agentes

Encaja especialmente con

- Desarrolladores que trabajan cada día con Claude Code o GitHub Copilot y notan que la herramienta rinde menos de lo que debería, o que cuesta más.
- Tech leads que quieren que el equipo trabaje con agentes de una sola forma, y no de una por persona.
- Engineering managers que quieren controlar el gasto en tokens sin frenar al equipo.
- Equipos de plataforma que mantienen la configuración, los permisos y los agentes del resto de la organización.
- Equipos que ya lanzan agentes en paralelo o en autónomo y empiezan a ver bucles, commits mezclados o pull requests que no se sostienen.

No es la formación adecuada si

- Tu equipo todavía no usa un asistente de código sobre un repositorio real.
- Buscas una introducción a los LLM o un curso de prompts.
- Quieres comparar herramientas para elegir una. El workshop parte de Claude Code y Copilot.
- Esperas una auditoría de tu código. En el último bloque cada uno trabaja sobre su repositorio, pero yo no lo reviso por vosotros.

QUIÉN IMPARTE EL WORKSHOP

De la ciberseguridad de BBVA a la IA de Santander

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*.

Lo que enseño aquí es lo que uso en mis proyectos. Cada uno tiene su índice de documentación, y reparto el trabajo entre agentes, cada uno con su modelo, su límite de turnos y su informe. Una conclusión importante no se da por buena hasta que otro agente la comprueba por otro camino.

GARANTÍAS

Dos compromisos por escrito y un límite

Si las configuraciones no funcionan, las arreglo

Si las configuraciones que dejo en vuestro repositorio no funcionan, hago una sesión remota gratis para arreglarlas.

Si os atascáis en 30 días, una hora gratis

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

Esto no es una auditoría ni te prometo resultados: no garantizo porcentajes de ahorro. El ahorro se mide durante el día, con vuestras sesiones.

ANTES DE RESERVAR

Lo que me suelen preguntar antes de reservar

¿Cuántos tokens vamos a ahorrar?
No os doy una cifra antes de ver vuestras sesiones. Durante el día cada participante mide su consumo antes y después de cada laboratorio, así que os vais con vuestro número, no con el mío.

¿Cuánto cuesta?
Depende de la duración y del tamaño del equipo. Lo hablamos en la primera conversación y te paso una propuesta cerrada.

¿Hace falta que todos usen la misma herramienta?
No. Cada participante necesita Claude Code o GitHub Copilot ya iniciado y listo para trabajar contra una carpeta local. Lo explico en las dos, y buena parte de la configuración sirve para ambas a la vez.

¿No es demasiado básico si ya los usamos a diario?
El workshop parte de ahí. Lo básico lo repaso rápido y el tiempo se va en lo que casi nadie configura: modelo por ronda, caché, límites de ejecución, bucle Ralph, hooks, paneles de agentes y configuración gestionada.

¿Podemos usar nuestro repositorio durante el día?
Sí, en el último bloque. Los laboratorios anteriores usan un repositorio de ejercicios para que todo el grupo vea el mismo fallo. Después cada uno aplica lo aprendido sobre su propio código, en su portátil.

¿Nuestro código sale de la empresa?
No. Se trabaja en vuestros portátiles y con vuestras cuentas. Yo no me llevo ni copio código, y lo que cada herramienta envía a su proveedor es lo mismo que envía cualquier otro día.

¿Se adapta a nuestro lenguaje o stack?
Los laboratorios usan Python, pero los conceptos valen para cualquier lenguaje. El último bloque se hace sobre el stack de cada participante, sea cual sea.

¿Se puede bonificar por FUNDAE?
Pregúntamelo al pedir fecha. Te paso lo que necesites para que vuestra gestoría o vuestro departamento de formación lo valoren.

¿Se puede hacer online?
Sí, con el mismo temario, los mismos laboratorios y la misma medición. Va en directo y el reparto en sesiones lo decidimos juntos. El bloque final en vuestro repositorio se hace compartiendo pantalla.

¿Y si necesitamos seguimiento?
Hay un programa de 6 semanas sobre lo mismo, con medición antes y después y revisión de vuestros PR. Tienes el detalle en el programa.

FORMATO

Un equipo, un día y una forma común de trabajar con agentes

Incluye la preparación, la jornada de cinco a ocho horas y todo el material de trabajo. Se puede hacer de dos formas. La duración y el tamaño del grupo los cerramos en la primera conversación, para que haya tiempo de repasar los laboratorios y de acompañaros en el bloque final, en el repositorio de cada uno.

Presencial

En vuestra oficina, todos en la misma sala.

- Sala con pantalla, wifi y un portátil por participante.
- Desplazamiento fuera de Madrid acordado antes de cerrar la fecha.

Online

En directo por videollamada, con el mismo programa y los mismos laboratorios.

- Cada participante con su portátil, cámara y micrófono.
- Sin desplazamientos, así que vale para equipos repartidos en varias ciudades.

- Participantes del mismo equipo o de equipos que trabajan juntos.
- Impartición en español o en inglés.
- Incluye los números del equipo con su página resumen, las configuraciones en vuestra rama y los dos compromisos de arriba.
- Si buscáis una auditoría o una charla para escuchar, esto no es lo vuestro. Es un día de trabajo para un equipo que va a aplicar lo aprendido.

Escríbeme a **yo@danielalfocea.com**

En la primera conversación vemos si encajamos, qué herramientas usa el equipo, cuántos sois, cuánto dura y qué fecha os viene bien. Con eso os paso la propuesta.

Bonificable a través de FUNDAE.

CORREO

yo@danielalfocea.com

PROGRAMA EN LA WEB

danielalfocea.com/formacion/workshop-productividad-con-ia