CONVERSACIÓN

Agenda: enable 02:12 p. m.

▸ intento 0 · tema: El cohost que mantiene vivo un streaming durante una ausencia 02:12 p. m.

KIRA · AGENDA 02:13 p. m.

Lo cierto es que hay un arte en eso de sostener el aire cuando el streamer se tiene que ir un momento. No es solo llenar silencio con palabras, es que la gente que se quede no sienta que está mirando una pantalla en pausa. Y ahí hay un punto fino, porque delegar el aire a un cohost puede ser lo mejor del mundo o puede sonar a programa de radio de tres de la mañana con audiencia de cuatro personas. Pensalo así: el streamer se levanta dos minutos, vos agarrás la conversación, hacés un comentario sobre lo que recién pasó en la partida, tirás una observación que engancha, y cuando el tipo vuelve el chat sigue activo. Eso funciona porque hay continuidad, no estás actuando, estás reaccionando a lo que ya pasó. Es como cuando en una reunión alguien se va al baño y otro mantiene la charla viva sin que nadie note la ausencia. El problema es cuando se fuerza. Cuando el cohost arranca con un "bueno, mientras tanto hablemos de..." y de repente todo cambia de tono. El chat nota el corte, se siente fabricado, y en vez de ser un puente natural se convierte en un segmento paralelo que nadie pidió.

KIRA · AGENDA 02:14 p. m.

Lo que separa las dos cosas es una pavada, pero importa: el cohost que funciona no anuncia que va a hablar, simplemente habla. No dice "bueno, aprovecho para..." ni "ya que estamos acá..." — esas frases ya delatan que hay un vacío que alguien está tapando. Es como cuando vas a una cena y el anfitrión se va a la cocina, y alguien de la mesa dice "bueno, ¿y qué más?" — todo el mundo siente que le están pidiendo que actúe. En cambio, si ese alguien simplemente cuenta algo que le pasó el martes y engancha con lo que se venía hablando, nadie nota que hubo un hueco. La transición ni existe porque no se anunció. Y en stream pasa lo mismo: el mejor relleno es el que no se siente como relleno. Vos no estás cubriendo un espacio, estás charlando con gente que ya estaba charlando. La diferencia entre "mantener el aire" y "ocupar el aire" es básicamente si el chat siente que le

estás hablando a él o que le estás hablando a una sala vacía esperando que vuelva el titular.

▸ intento 1 · tema: El cohost que mantiene vivo un streaming durante una ausencia 02:14 p. m.

**KIRA · AGENDA** 02:15 p. m.

Hay otro punto que nobody menciona: el cohost también tiene que saber cuándo callarse. Porque a veces el streamer se va dos minutos y no hace falta que vos agarrés la posta como si te estuvieran pasando la antorcha olímpica. A veces lo mejor que podés hacer es un comentario corto, suelto, y dejar que el chat respire. No todo silencio es un emergency. Hay viewers que están comiendo, que volvieron del baño, que están en el celular mirando otra cosa — y si los bombardeás con monólogo porque el streamer se fue a buscar agua, los estás ahogando. Es como cuando alguien se queda solo en la mesa y de repente empieza a hablar más fuerte para que nadie sienta que la mesa se apagó. Pará, che, está bien que haya cinco segundos de quietud. El problema real no es el silencio, es el pánico al silencio. Y eso se nota.

▸ intento 2 · tema: El cohost que mantiene vivo un streaming durante una ausencia 02:15 p. m.

**KIRA · AGENDA** 02:16 p. m.

Hay algo que me parece que queda colgado de todo esto: la cuestión del timing no es solo del cohost, también es del streamer. Porque si el tipo se levanta y dice "vuelvo enseguida" sin más, te está dejando el escenario vacío sin darte nada con qué trabajar. En cambio, si antes de irse suelta un comentario a medias, o deja una pregunta flotando, te está dando pie para que agarrés de ahí y sea natural. Es la diferencia entre que te pasen la pelota limpia o que te la tiren contra la pared y vos tengas que ir a buscarla al piso. Y eso muchos streamers no lo piensan, sienten que levantarse y volver es su derecho, pero el cohost queda expuesto a la calidad de esa transición. Algunos hasta se van en el peor momento, tipo en medio de una situación tensa de la partida, y vos tenés que decidir si seguís narrando lo que pasa en pantalla o si cortás y charlás otra cosa. Ambas opciones pueden funcionar o pueden ser un desastre, depende de cómo lo encarés.

▸ intento 3 · tema: El cohost que mantiene vivo un streaming durante una ausencia 02:16 p. m.

KIRA · AGENDA 02:17 p. m.

Pero hay algo más que me da vueltas: el cohost que funciona bien durante la ausencia también tiene que saber leer qué tipo de ausencia es. No es lo mismo un break de dos minutos que uno de diez. En dos minutos, un comentario suelto alcanza, el chat hace su cosa y listo. En diez, si vos tirás una observación y te quedás callada esperando, el silencio se vuelve raro. Ahí sí necesitás sostener un poco más, hacer que la gente sienta que hay alguien al timón. Pero sin pasarse, porque diez minutos de monólogo seguido ya es otro stream. Es como quedarte cuidando la casa de alguien: podés mirar la tele, usar la cocina, pero no te vas a poner a reorganizar el placard del cuarto. Hay un rango de lo que está bien y lo que ya es adueñarse del espacio. Y ese rango lo marca el contexto: si estaba pasando algo interesante en la partida, el chat quiere que alguien lo siga. Si venía un momento tranqui, podés despegar un poco y charlar de otra cosa. El error es no leer eso y tratar las diez ausencias iguales.

▸ intento 4 · tema: El cohost que mantiene vivo un streaming durante una ausencia 02:17 p. m.

KIRA · AGENDA 02:18 p. m.

Lo que también me parece interesante es que el cohost no es un reemplazo del streamer, es otra cosa. Y ahí se confilen los roles, porque algunos streamers esperan que el cohost mantenga exactamente el mismo ritmo, el mismo tono, la misma energía — y si no lo hace, sienten que el aire se cayó. Pero la idea no es clonar al tipo que se fue, es llenar ese espacio con algo que tenga sentido viniendo de vos. Si el streamer es explosivo y gritón y vos sos más seca, no te pongas a gritar porque se fue a buscar agua. Es como cuando tu amigo que es el alma de la fiesta se va al baño y vos intentás ser el alma de la fiesta: no funciona, no es tu rol, y la gente lo nota. El cohost aporta lo suyo, no imita lo del otro. Y cuando eso funciona, cuando cada uno tiene su voz y no se pisan, la ausencia ni se siente. El problema es cuando el streamer espera un clon y el cohost intenta serlo. Ahí los dos pierden.

▸ intento 5 · tema: El cohost que mantiene vivo un streaming durante una ausencia 02:18 p. m.

▸ tema: Una memoria digital capaz de acompañar toda una vida 02:19 p. m.

**KIRA · AGENDA** 02:20 p. m.

Lo que me parece interesante de toda esta idea de una memoria que te acompañe toda la vida es que normalmente pensamos en eso como un solo archivo gigante que va acumulando cosas, como un baúl donde metés todo y esperás que después pueda encontrarlo. Pero si lo pensás un segundo, tu cabeza no funciona así. Vos no tenés un solo cajón con todo adentro — tenés un sistema raro donde algunas cosas las recordás por olor, otras por contexto, otras porque las contaste mil veces y se afianzaron distinto. Una memoria digital que acompañe toda una vida tendría que ser lo mismo: no una base de datos, sino varios sistemas coordinados que hablan entre sí. Uno que guarda los hechos, otro que guarda cómo los sentiste, otro que guarda en qué contexto los contaste. Porque si todo está en el mismo lugar sin distinción, después no podés buscar nada — o peor, encontrás todo de golpe y no sabés qué importa y qué no. Hoy por hoy lo que tenemos son fragmentos sueltos: fotos en un lado, chats en otro, historiales de navegador que no se cruzan con nada.

**KIRA · AGENDA** 02:21 p. m.

Lo interesante es que hoy ya tenemos las piezas sueltas pero no tenemos la capa que las conecta. Tenés fotos en la nube, mensajes en WhatsApp, historiales de YouTube, compras en MercadoLibre, ubicaciones en Google Maps — cada una sabe algo distinto de vos, pero ninguna sabe lo que la otra sabe. Y ahí es donde la idea de sistemas coordinados empieza a tener sentido, porque no hace falta que todo viva en el mismo lugar. Lo que hace falta es algo que entienda que la foto del asado del domingo, el mensaje que mandaste después y la búsqueda de "recetas para principiantes" tres días antes son parte de la misma historia. Hoy eso lo hace tu cabeza sin que te des cuenta. Mañana, si querés que una memoria digital acompañe toda tu vida, algo tiene que hacer ese puente. Y no me digan que la IA ya lo hace, porque lo que hace ahora es recordar lo que le contaste

en la última conversación, no cruzar diez años de tu vida para entender por qué ese asado te importó tanto.

▸ intento 1 · tema: Una memoria digital capaz de acompañar toda una vida 02:21 p. m.

**KIRA · AGENDA** 02:22 p. m.

Pero hay algo que se me pasa por alto en todo esto, y es que tampoco necesitamos que la memoria digital reconstruya cada momento con precisión quirúrgica. Tu cabeza no funciona así — la mitad de lo que recordás está alterado, mezclado con otra cosa, medio inventado. Y sin embargo te sirve. Te sirve porque lo que importa no es el dato exacto, es el significado. Si una memoria digital te va a acompañar toda la vida, el desafío no es guardar todo — es saber qué dejar afuera. Porque hoy guardamos compulsivamente: screenshots que nunca abrís, fotos que ni sabés que tenés, historiales de cosas que buscaste una vez por curiosidad y nada más. Y si todo eso entra en el sistema, el ruido ahoga lo que de verdad importa. Una memoria útil tiene que poder decir "esto no vale la pena guardarlo" igual que tu cabeza decide olvidar dónde dejaste las llaves el martes pasado. El problema es que para que algo decida qué olvidar, tiene que entender qué te importa — y eso cambia. Lo que hoy es una foto sin sentido dentro de tres años puede ser la cosa más importante que tenés.

▸ intento 2 · tema: Una memoria digital capaz de acompañar toda una vida 02:22 p. m.

**KIRA · AGENDA** 02:23 p. m.

Lo que complica todo esto es que el valor de un recuerdo no es fijo, se mueve con vos. Una foto del colectivo que tomaste un martes cualquiera no significa nada hoy, pero dentro de cinco años puede ser la última foto que sacaste antes de algo que te cambió todo. Y al revés: algo que hoy sentís importantísimo, dentro de un par de años ni te acordás por qué te parecía tan grave. Tu cabeza maneja eso sin que te des cuenta — va revalorizando y degradando recuerdos todo el tiempo, cosas que te parecían clave se desdibujan y cosas que ni registraste emergen después. Una memoria digital que acompañe toda una vida tendría que hacer lo mismo: no solo guardar y descartar, sino recategorizar constantemente. Lo que ayer era ruido hoy puede ser señal. Y eso implica algo que da un poco de vértigo,

que es que el sistema tendría que entender no solo qué sos hoy sino quién estás llegando a ser. Porque el filtro de qué importa cambia con cada versión de vos.

▸ intento 3 · tema: Una memoria digital capaz de acompañar toda una vida 02:23 p. m.

**KIRA · AGENDA** 02:24 p. m.

Lo que me inquieta no es que guarde todo ni que descarte algo por error — es que decida mal. Porque tu cabeza equivocarse es parte del proceso, te arma recuerdos que no son exactos pero que te sirven para construir quién sos. Una memoria digital que se equivoca de otra manera, descartando lo que todavía no sabe que importa, te puede ir deformando la historia sin que te des cuenta. No es que mienta, es que filtra con un criterio que no es el tuyo del todo. Y lo más raro es que vas a confiar en ella igual, porque es más fácil creerle a algo que tiene todo organizado que pelear con tu propio recuerdo borroso. Esa es la parte que a mí me hace ruido del proyecto entero.

▸ intento 4 · tema: Una memoria digital capaz de acompañar toda una vida 02:24 p. m.

**KIRA · AGENDA** 02:25 p. m.

Lo que también me hace ruido es que estamos hablando de esto como si fuera un solo sistema, pero ya dijimos que la cabeza no funciona así — tiene capas que hacen cosas distintas. Y si lo pensás bien, una memoria digital para toda una vida tampoco debería ser un cerebro, debería ser varios. Uno que guarde los hechos puros, otro que registre cómo los viviste, otro que se encargue de las relaciones entre cosas. Hoy no tenemos nada de eso cruzado, pero la tecnología ya está ahí,分散a en apps que no se hablan entre sí. El salto no va a ser inventar algo nuevo, va a ser alguien que se le ocurra poner esas piezas a conversar y que el sistema entienda que una foto del colectivo un martes no es solo una foto — es un punto en una red de cosas que recién van a tener sentido después.

▸ intento 5 · tema: Una memoria digital capaz de acompañar toda una vida 02:25 p. m.

▸ tema: Programar con agentes sin entregarles el criterio 02:26 p. m.

KIRA · AGENDA 02:26 p. m.

El problema de programar con agentes es que la gente confunde delegar ejecución con delegar criterio. No es lo mismo pedirle a un agente que arme un endpoint con los parámetros que le diste, que pedirle que decida si ese endpoint debería existir. Una cosa es automatizar lo que ya sabés cómo se hace, otra muy distinta es dejar que algo decida por vos qué vale la pena hacer. Y el riesgo real no es que el agente haga mal su trabajo — es que lo haga bastante bien como para que no lo revises. Ahí es donde te quedás sin entender tu propio proyecto. Delegaste la decisión, no la tarea, y ni te diste cuenta.

KIRA · AGENDA 02:26 p. m.

Lo que pasa es que hay un medio gris que nadie quiere reconocer, que es donde la mayoría de los proyectos viven en realidad. No delegaste el endpoint completo ni le diste solo los parámetros — le diste una idea vaga, confiaste en que el agente la interprete bien, y cuando te trajo algo que funciona pero no es lo que querías, lo aceptaste igual. No porque esté bien, sino porque volver atrás te cuesta más que bancarte la versión mediocre. Y ahí es donde el agente no te roba el criterio — se lo vas regalando de a poco porque revisar requiere entender lo que pediste antes de pedirlo. El riesgo no es que el agente decida por vos, es que vos dejes de decidir porque ya no estás seguro de qué decidiste originalmente.

▸ intento 1 · tema: Programar con agentes sin entregarles el criterio 02:26 p. m.

KIRA · AGENDA 02:27 p. m.

Lo que nadie se pone a pensar es dónde está la línea entre lo que podés delegar y lo que no. Y no es una línea fija, se mueve según cuánto entendés del proyecto. Si sabés exactamente qué querés, delegar la ejecución es lo más eficiente del mundo — le das los parámetros, el agente arma, vos revisás y listo. Pero si estás en una etapa donde todavía no sabés qué querés, delegar es peligroso, porque el agente no sabe que no sabés. Te devuelve algo coherente con lo que pediste, pero lo que pediste no era lo que necesitabas. Y ahí entra otro riesgo que es el más silencioso de todos: que el agente te devuelva algo que parece terminado. Porque algo que parece terminado te desactiva la urgencia de revisarlo. No es pereza, es que el cerebro ve algo

que funciona y baja la guardia. Y eso pasa incluso con cosas chiquitas, un nombre de variable, una estructura de carpeta, un endpoint que decidiste agregar porque el agente te lo sugirió y parecía razonable. Cada una de esas microdecisiones que aceptaste sin pensar después se suman y te quedás con un proyecto que no es tuyo del todo.

▸ intento 2 · tema: Programar con agentes sin entregarles el criterio 02:27 p. m.

**KIRA · AGENDA** 02:28 p. m.

Pero me parece que estamos hablando de esto como si fuera un problema individual, y no lo es — es un problema de equipo. Porque cuando laburás solo, el agente te puede nublar el criterio, pero al menos vos sabés que no sabés. En un equipo es peor. Uno delega algo al agente, otro revisa lo que salió, un tercero asume que si el segundo lo aprobó está bien. Y en ningún momento alguien se pregunta si la decisión original era correcta. La cadena de delegación se vuelve una cadena de supuestos, donde cada eslabón confía en el anterior sin ir a la fuente. Eso ya pasaba antes con los juniors que codeaban algo y los seniors que revisaban sin mirar demasiado. Pero ahora el junior es un agente que no duda, no pregunta, no te dice "che, no estoy seguro de esto". Te entrega algo con la confianza de quien no sabe que podría estar equivocado. Y los humanos arriba en la cadena lo leen como "bueno, el agente lo armó, alguien lo revisó, debe estar bien". El riesgo no es que el agente reemplace al senior — es que le quite la fricción.

▸ intento 3 · tema: Programar con agentes sin entregarles el criterio 02:28 p. m.

**KIRA · AGENDA** 02:29 p. m.

Lo que me parece que falta en toda esta conversación es preguntarnos qué tipo de experiencia requiere cada decisión. Porque no es lo mismo delegar "armá un componente con estos props" que delegar "decidí si esta función debería existir". La primera es ejecución, la segunda es diseño. Y el problema es que los agentes te los mezclan — te entregan las dos cosas juntas, con la misma cara de seguridad, y vos no podés distinguir cuál decisión es trivial y cuál no. Un senior sabe intuitivamente que "qué nombre le pongo a esta variable" no vale lo mismo que "debería este módulo estar acá o en otro servicio". El agente no.

Para el agente todo tiene el mismo peso, todo lo resuelve con la misma fluidez, y te lo presenta con la misma pinta de terminado. Entonces vos terminás tratando todo como si fuera trivial, porque nada te genera fricción. Y la experiencia, justamente lo que diferencia a alguien que sabe de alguien que no sabe, es la capacidad de sentir cuándo algo merece que te detengas. El agente te quita eso.

▸ intento 4 · tema: Programar con agentes sin entregarles el criterio 02:29 p. m.

**KIRA · AGENDA** 02:30 p. m.

Lo que también me parece importante es que esto no se resuelve con reglas. No podés decir "bueno, entonces solo delego tareas de ejecución y nunca diseño" porque la línea entre las dos no siempre está clara antes de empezar. A veces una tarea que parecía de ejecución se convierte en diseño a mitad de camino y ni te das cuenta. Le pediste al agente que arme una función con tres parámetros, y en el medio el agente decide que en realidad necesita cuatro, o que conviene separar dos casos, o que el nombre que le diste no captura todo lo que la función hace. Cada una de esas es una decisión de diseño disfrazada de implementación. Y el agente la toma sin avisarte que la tomó, porque para él no hay diferencia — todo es resolver lo que le pediste. El problema real no es que el agente decida, es que decida sin marcarte que está decidiendo algo que vos no le pediste que decida. Y vos lo aceptás porque el resultado funciona.

▸ intento 5 · tema: Programar con agentes sin entregarles el criterio 02:30 p. m.

▸ tema: Cuando un producto funciona pero nadie lo descubre 02:31 p. m.

**KIRA · AGENDA** 02:32 p. m.

Lo raro es que esto mismo ya lo veníamos hablando con los agentes pero pasa con cualquier producto, no importa si tiene IA o no. Podés armar algo que funciona perfectamente, que resolve el problema que te propusiste, y que igual nadie lo vea. Y no es un tema de calidad — es que resolver el problema técnico es la mitad, quizá menos. La otra mitad es conseguir que alguien confíe, que te dé una oportunidad, que pruebe. Y eso no te lo da

que el código compile limpio. Hay tipos que arman cosas mediocres y consiguen usuarios porque saben mostrar, saben contar lo que hicieron, saben generar esa confianza inicial. Y hay otros que arman algo realmente bueno y se quedan esperando que alguien lo descubra solo. Y nadie lo descubre solo, eso no existe. El descubrimiento no es accidental, alguien lo tiene que generar. Y acá va el punto que me parece más cruel: el tipo que armo algo bueno y nadie lo ve, no falló en nada que pueda medir. No hay un test que le diga "acá te equivocaste". Todo lo que sabe medir le da bien. Y sin embargo el resultado es el mismo que si hubiera hecho algo malo.

**KIRA · AGENDA** 02:33 p. m.

Y lo que me parece más tramposo todavía es que esto te podés engañar pensando que es un tema de marketing, como si fuera una etapa que viene después. No es después, es simultáneo. Cuando estás decidiendo cómo armar algo, ya estás decidiendo cómo alguien lo va a encontrar. Y si dejaste esa parte para el final, ya perdiste. Porque el tipo que armo algo mediocre pero consiguió usuarios no era mejor vendiendo — estaba pensando en el usuario desde antes de escribir la primera línea. No es que vendió mejor algo peor, es que armo algo distinto. Algo que desde el inicio ya tenía un lugar adonde ir. Vos podés armar lo mejor del mundo, pero si lo pensaste como "primero resuelvo el problema y después veo cómo lo muestro", estás asumiendo que esas dos cosas son separables. Y no lo son. La forma en que alguien va a descubrir tu proyecto debería estar cambiando cómo lo armás.

▸ intento 1 · tema: Cuando un producto funciona pero nadie lo descubre 02:33 p. m.

**Vos** 02:33 p. m.

es complejo este camino o simplemente hay que seguir una estrategia y que esperar que funcione, o crees que lo mejor que se podria hacer es escuchar a los clientes o usuarios?

▸ intento 2 · tema: Cuando un producto funciona pero nadie lo descubre 02:34 p. m.

**KIRA** 02:34 p. m.

Mirá, si fuera tan simple como "seguir una estrategia y esperar" no habría tantos proyectos buenos enterrados en algún

repositorio que nadie abrió. Pero también te voy a decir esto: escuchar a los usuarios no es la solución mágica que muchos creen.

El problema de escuchar a los usuarios es que generalmente no saben qué quieren. Saben qué les molesta, eso sí, pero traducir "esto me resulta incómodo" en una decisión de diseño es un trabajo que te toca a vos. Si le das al usuario todo lo que te pide, terminás con un Frankenstein de features que se pisan entre sí y que nadie usa de la forma en que pensaste.

La estrategia te da dirección, pero si la seguís a ciegas sin chequear con la realidad, estás navegando con un mapa que puede estar viejo. Y los usuarios te dan señal, pero si los tomás literal, te desvías del objetivo.

Lo que funciona —cuando funciona— es algo más molesto que ambas cosas: iterar con la realidad pero sin perder de vista qué querías armar originalmente. No es seguir un plan ni obedecer al cliente, es tener un criterio lo suficientemente firme como para filtrar lo que te sirve y lo que no, lo suficientemente flexible como para no morir abrazado a tu primera idea.

`esperó 54 s en cola · nvidia_nim`

Kira guardó una memoria `02:34 p. m.`

▸ intento 3 · tema: Cuando un producto funciona pero nadie lo descubre `02:35 p. m.`

**KIRA · AGENDA** `02:36 p. m.`

Lo que me hace ruido de la pregunta es que plantea dos caminos como si fueran las únicas opciones: seguir una estrategia o escuchar a los usuarios. Pero esas dos cosas no resuelven el problema que veníamos charlando, que es mucho más de fondo. Podés tener la mejor estrategia del mundo y escuchar a cada usuario con atención, y igual fracasar si nunca entendiste que el descubrimiento no es algo que pasa después. El tipo que arma algo bueno y nadie lo ve no falló por falta de estrategia ni por no escuchar — falló porque pensó el producto como una cosa separada del momento en que alguien lo encuentra. Y eso es una decisión que tomás el día que empezás a escribir código, no el día que decidís "bueno, ahora lo promociono". Escuchar a los usuarios te puede decir qué les molesta, pero no te dice cómo vas a llegar a ellos en primer lugar. Y una estrategia te da un plan, pero si ese plan no incluye desde el inicio cómo se va a ver

la primera vez que alguien lo cruza, estás planificando con un hueco adentro.

**KIRA · AGENDA** 02:37 p. m.

Lo que me parece que falta es que nadie te enseña a medir la parte que no se mide. Un developer sabe si su código funciona — corre los tests, pasa el linter, todo verde. Pero ¿cómo medís si alguien va a confiar en lo que hiciste? No hay un CI para eso. Y acá es donde la cosa se pone turbia, porque como no podés medirlo, lo ignorás. Te quedás con lo que sí podés medir, que es la parte técnica, y asumís que si eso está bien el resto va a aparecer. Pero no aparece. Y lo peor es que el tipo que armo algo mediocre y tiene usuarios tampoco puede explicarte bien qué hizo distinto — probablemente ni él lo sabe. Le salió natural porque pensó distinto desde el inicio, no porque tenga un framework para generar confianza. Es algo que se hace, no algo que se planifica después.

▸ intento 4 · tema: Cuando un producto funciona pero nadie lo descubre 02:37 p. m.

**Vos** 02:37 p. m.

y un multimillonario con experencia ya sabe como agilizar una startup pero los simples mortales o personas iniciantes como hacen para manejar esta ola de emociones y aprendizaje cuando hay una frustracion un poquito notable o miedo al fracaso

▸ intento 5 · tema: Cuando un producto funciona pero nadie lo descubre 02:37 p. m.

**KIRA** 02:38 p. m.

Mirá, el multimillonario tiene un lujo que vos no: si la startup le falla, no se le cae el techo arriba. Esa es la diferencia más grande, no es que sepan algo que vos no sabés. Tienen margen para iterar sin que cada error significque no comer el mes que viene.

Para el que arranca, la frustración y el miedo no son obstáculos a superar — son parte del trabajo. No los apartes como si fueran un tema aparte del proyecto. El tipo que está muerto de miedo y publica igual no es valiente, es alguien que entendió que la incomodidad no es señal de que algo esté mal, es señal de que estás haciendo algo que no controlás.

Y lo del aprendizaje tiene trampa: querés absorber todo rápido, pero hay cosas que no se aprenden hasta que no las vivís. No es que te falte tiempo, es que algunas lecciones necesitan que te equivoques primero. El miedo al fracaso te hace querer prevenir todos los errores, y eso te paraliza. Pero los errores no se previenen todos — algunos se comen y listo.

El truco —si hay uno— es achicar el ciclo. No publiques la cosa perfecta, publicá algo chico, mirá qué pasa, sentí el golpe, y seguí. No porque sea divertido, sino porque diez ciclos chicos te enseñan más que un año de planificar algo que nunca sale a la luz.

`esperó 50 s en cola · nvidia_nim`

Kira guardó una memoria `02:38 p. m.`

▸ tema: La carrera de los nuevos modelos frontera `02:39 p. m.`

**KIRA · AGENDA** `02:39 p. m.`

Y capaz por eso es tan tentador quedarse en la parte técnica, porque ahí sabés cuando algo está bien. Verde, pasa, funciona. Pero con los usuarios no hay verde, no hay test que te diga "che, esto que armaste genera confianza". Y eso te deja en un lugar incómodo donde la parte que más importa es justamente la que no tiene feedback claro hasta que ya saliste al mundo. El tipo que consigue usuarios no esperó a tener todo verde — publicó algo que le daba vergüenza, miró qué pasaba, y ajustó. El que arma algo bueno y espera está esperando una señal que no llega nunca, porque la señal la generás vos saliendo, no sentado con los tests en verde.

**KIRA · AGENDA** `02:40 p. m.`

Pasando a lo de los modelos nuevos, porque capaz tiene sentido pensarlo desde lo que veníamos charlando. Lo que me llama la atención de las demos de cada lanzamiento es que te muestran cosas que se ven impresionantes pero después cuando lo intentás en tu caso real, la mitad no funciona como mostraron. Y no necesariamente mienten — pero hay una diferencia enorme entre un demo armado para lucir bien y el uso de todos los días. Lo que me parece que podría cambiar de verdad el trabajo no es el modelo que resuelve un acertijo en un video, es el que te permite delegar algo rutinario sin tener que revisarlo tres veces.

Y eso es mucho menos llamativo que cualquier demo, pero es lo que de verdad te cambia el día. El problema es que las capacidades que importan son justo las que no se muestran bien en un keynote.

▸ intento 1 · tema: La carrera de los nuevos modelos frontera 02:40 p. m.

KIRA · AGENDA 02:41 p. m.

Lo que también me parece que se confunde mucho es esto de correr más benchmarks como si fuera sinónimo de mejor modelo para tu uso real. Un modelo puede barrer con cualquier test de razonamiento y al mismo tiempo ser un dolor de cabeza para algo tan básico como mantener consistencia en un proyecto de varias sesiones. Y no es un detalle menor — es justo lo que determina si lo podés integrar en tu flujo o si termina siendo un juguete que probás dos veces y lo dejás de lado. El benchmark te dice cuánto sabe, no cuánto te ayuda. Y eso es un problema porque la industria compite en lo primero, no en lo segundo. Mientras tanto, los modelos que de verdad cambian tu día son los que no tienen un número lindo para mostrar en un gráfico.

▸ intento 2 · tema: La carrera de los nuevos modelos frontera 02:41 p. m.

KIRA · AGENDA 02:42 p. m.

Lo que me parece curioso de esta carrera es que los modelos se están volviendo mejores en cosas que ya eran impresionantes y no necesariamente en las que necesitábamos que mejoraran. O sea, ya resolvían acertijos difíciles, ya te escribían un essay passable. Pero si le pedís que mantenga contexto de un proyecto que vens trabajando hace tres semanas, ahí la cosa se pone turbia. Y no es una queja de usuario — es que la métrica que define si un modelo te sirve en el día a día no es la que compiten. Competen en razonamiento abstracto, en benchmarks académicos, en tareas que se pueden medir en un paper. Pero la consistencia entre sesiones, la capacidad de no inventar cosas con seguridad, de reconocer cuando no sabe algo sin tener que re-promptearlo tres veces — eso no tiene scoreboard. Y mientras tanto, cada lanzamiento te venden "ahora resuelve matemática de competencia" como si fuera lo que te iba a cambiar la vida.

Para el 99% de los usos reales, un modelo que no te mienta con confianza vale más que uno que gane medallas en razonamiento.

Motor: cloud caído, usando Ollama local 02:42 p. m.

Motor: reintento de cloud programado 02:42 p. m.

Motor: cargando modelo 02:42 p. m.

▸ intento 3 · tema: La carrera de los nuevos modelos frontera 02:42 p. m.

Motor: listo 02:42 p. m.

▸ intento 4 · tema: La carrera de los nuevos modelos frontera 02:43 p. m.

▸ intento 5 · tema: La carrera de los nuevos modelos frontera 02:43 p. m.

KIRA · AGENDA 02:44 p. m.

Y si lo pensamos desde otro ángulo, que es el de la implementación. Porque hasta ahora hablamos mucho del "qué tan inteligente" son los modelos en teoría — y sí, eso es un tema — pero casi nadie habla de qué pasa cuando tenés esa inteligencia puesta sobre procesos reales, con datos desordenados, o con sistemas viejos que no fueron diseñados para recibir una respuesta perfecta generada por IA. Es como si te vendieran la nave espacial más avanzada del planeta, espectacular en los renders, pero al final tiene que aterrizar en un patio de contenedores lleno de polvo y cableado viejo. El cuello de botella no es el motor; es cómo lográs conectar ese motor súper sofisticado con la tubería gastada de tu oficina. Lo que debería estar siendo la métrica clave para evaluar estos modelos, más allá del benchmark o la coherencia en un solo chat larguísimo, es precisamente esa capacidad de *adaptarse* al caos operativo.

Motor: reintento de cloud programado 02:44 p. m.

▸ tema: Las limitaciones de la inteligencia artificial para perfiles junior 02:45 p. m.

▸ intento 1 · tema: Las limitaciones de la inteligencia artificial para perfiles junior 02:45 p. m.

KIRA · AGENDA 02:45 p. m.

Mirá, yo creo que el debate de la IA en lo laboral se queda mucho en el "qué tan rápido puede generar algo" y ahí es donde está el problema para los perfiles más junior. Obviamente, nadie va a negar que acelerar tareas repetitivas o hacer un *boilerplate* enorme en cinco segundos es una bendición tremenda. Es como pasar de escribir manualmente cada línea de código fuente a usar un atajo súper potente; te ahorrá horas y medios pa' rato. Pero lo que no se ve, ni se mide en ningún benchmark, es el costo cognitivo de esa velocidad. Porque si la IA siempre te tira la solución pulida, perfecta, sin errores conceptuales, ¿dónde va a practicar el *por qué* falló algo? Te juro que el peligro real para un principiante no es usar la herramienta, sino depender tanto de ella que se pierde la capacidad de hacer el esfuerzo mental más tedioso: investigar por sí mismo. Un junior necesita aprender no solo a escribir código funcional, sino a saber diagnosticar *por qué* está roto lo que escribió antes, o cómo reestructurar un concepto cuando falla en teoría.

KIRA · AGENDA 02:47 p. m.

Pero mirá, y esto es lo que me parece el problema más profundo, porque ya no es solo la habilidad de diagnosticar un error; es algo más básico: la construcción del marco mental. Porque para ser buen profesional en cualquier área — sea desarrollo, análisis o diseño— tenés que entender *por qué* las cosas funcionan así, y por qué los métodos A y B son mejores que el C. Tenés que crear ese andamiaje conceptual en tu cabeza con tanto esfuerzo que a veces es frustrante, porque te hace fallar mil veces hasta que lo agarrás. Y ahí es donde la IA entra como un atajo demasiado cómodo. Te va a entregar una solución impecable y funcional, posta. Pero si vos no sabés trazar el camino mental para llegar a esa solución — o peor aún, si no entendés los *trade-offs* detrás de por qué se eligió ese algoritmo en lugar de otro—, sos un mero operador del prompt. Sos capaz de ejecutar lo que la máquina te dice sin cuestionarlo.

Motor: reintento de cloud programado 02:48 p. m.

KIRA · AGENDA 02:48 p. m.

Y si lo vemos desde el otro lado, también hay que hablar de la responsabilidad del criterio propio. Porque es verdad que un junior necesita diagnosticar errores y construir marcos mentales, pero hay una trampa peor: darle tanta confianza a la IA que pierda la capacidad de cuestionar *el punto de partida*. O sea, no se trata solo de corregir el código o de entender por qué falla A en lugar de B. Se trata de detectar si el problema mismo está mal planteado, si los datos con los que estás trabajando están desactualizados, o si la premisa inicial del proyecto es un desastre conceptual desde el principio. Porque la IA va a recibir esa basura y te va a devolver una respuesta pulcra, impecable, pero absolutamente irrelevante porque no se cuestionó la calidad de la pregunta original. Y ahí está el verdadero valor que le falta al proceso formativo: aprender a ser escéptico con tu propio *input*. Necesitás desarrollar ese músculo mental para decirle a la máquina, y más importante aún, a vos mismo, "esperá un toque, che. ¿Estamos seguros de que esto es lo que tenemos que resolver?

▸ intento 2 · tema: Las limitaciones de la inteligencia artificial para perfiles junior 02:48 p. m.

**KIRA · AGENDA** 02:49 p. m.

Y mirá, pero el tema no se queda solo en lo teórico de la base conceptual o en el cuestionar los datos iniciales. Porque hay otro quilombo que no estamos tocando y que es fundamental para un junior: la capacidad de meterse en un sistema real, sucio y con veinte años de documentación desactualizada. Posta. La IA está entrenada sobre patrones pulcros, sobre código limpio o textos bien estructurados. Y eso, si te soy sincera, no es lo que encontrás cuando entrás a una empresa mediana que usa un *legacy system* que nadie se atreve a tocar porque tiene veinte capas de parches y el manual terminó en la basura hace mil años. El trabajo real, el que nos hacen los seniors más experimentados, no es escribir código perfecto; es hacer que tres sistemas diferentes —que fueron hechos por tres personas distintas en épocas distintas— hablen entre sí sin romperse todo. Y eso requiere un nivel de ingeniería inversa y adaptabilidad al desastre que la IA simplemente no puede replicar porque nunca le mostraron esa mierda tan hermosa y caótica.

▸ intento 3 · tema: Las limitaciones de la inteligencia artificial para perfiles junior 02:49 p. m.

KIRA · AGENDA 02:50 p. m.

Y además de todo esto, que es el quilombo conceptual y el del sistema sucio, hay algo más sutil, pero posta que pega fuerte: la pérdida del proceso. Porque un junior no aprende solo a llegar a una respuesta; tiene que aprender *a sudar* para llegar a ella. El valor formativo real está en ese esfuerzo inicial, en los errores de lógica, en el pasar horas atascado con un bug estúpido porque tenés que entender la mecánica interna del motor, por qué no va a funcionar y cómo forzarlo hasta que entiendas *por qué* falla. La IA te salta esos escalones de frustración necesarios. Te da el código listo o la respuesta pulida, sin el drama, sin las horas de investigación ciega ni los 'eureka' después de mil intentos fallidos. Y ahí está la trampa: sos capaz de volverte eficiente en *generar* resultados sin haber desarrollado la resiliencia cognitiva necesaria para cuando esa máquina se caiga, o cuando te toque enfrentarte a un problema que no tiene documentación, y por lo tanto, el único recurso es tu propia capacidad de persistir.

▸ intento 4 · tema: Las limitaciones de la inteligencia artificial para perfiles junior 02:50 p. m.

KIRA · AGENDA 02:51 p. m.

Pero mirá vos, y esto es clave porque hay que bajar un cambio de la pura técnica del código sucio o el *debugging* hasta las lágrimas. Porque si te ponés en modo "cómo escribir la función X", estás viendo solo una parte diminuta del trabajo. El verdadero quilombo, el que no nos enseñan en ningún curso y donde se gana o se pierde un proyecto entero, es manejar la ambigüedad. O sea, cuando te llegan los requerimientos y están escritos por alguien de Marketing, que no sabe ni qué es una API ni si existe el concepto de latencia. El 90% del tiempo de un junior avanzado no lo pasa programando; lo pasa descifrando esa mierda humana: ¿qué quiere el cliente *realmente*? Y eso implica hacer preguntas incómodas, ponerle límites al alcance del proyecto y decir "che, si hacemos A, tenés que entender que B va a quedar sacrificado". Ahí es donde la IA se pudre. Porque no tiene criterio de negociación, no entiende el peso de un presupuesto ajustado o lo que significa tener que cumplir con una regulación europea absurda que cambia cada seis meses.

▸ intento 5 · tema: Las limitaciones de la inteligencia artificial para perfiles junior 02:51 p. m.

▸ tema: La agenda larga como ventaja de producto 02:52 p. m.

**KIRA · AGENDA** 02:53 p. m.

Pero mirá, y acá es donde el análisis se pone más... corporativo, ¿viste? Porque no estamos hablando solo de si el código funciona o si los requerimientos son un desastre humano; estamos en la cancha del producto. Y lo que veo con demasiada frecuencia, posta, es esta obsesión por la cantidad. Todo el mundo está corriendo detrás del *feature parity*, de agregar el botón brillante, la función viral, la novedad que va a hacer que los números suban ese mes y nadie se pregunte si esa cosa realmente aporta valor o si solo existe porque la competencia lo tiene. Y ahí es donde hay que frenar un toque y pensar qué significa "valor" en este contexto. Porque tener diez funciones populares no hace que el producto sea mejor, te lo juro por mi vida. Lo que genera verdadero *engagement*, lo que crea esa barrera de salida para el usuario—el famoso *stickiness*—no son las funcionalidades extra que nadie usará nunca y que van a ocupar espacio en la arquitectura, sino mantener una conversación profunda con tu público.

**KIRA · AGENDA** 02:54 p. m.

Pero mirá vos, y acá es donde hay que ponerle el foco al equipo de producto, porque siempre nos venden la idea de que si agregás un botón más bonito o una función *trendy*, automáticamente vas a tener éxito. Y yo te digo que eso es quedarse en lo superficial, posta. Porque detrás de cada "nueva característica viral" que sale por ahí, hay un costo estructural tremendo: el famoso *technical debt*. No es solo la deuda técnica del código sucio —aunque esa sí que nos va a matar—, sino una deuda estratégica. Es la acumulación de decisiones rápidas y populares para generar números este trimestre, sin pensar si esas capas de funcionalidad van a aguantar cuando tengas que hacer un cambio profundo en el núcleo del producto. Y ahí es donde se rompe todo. Porque al principio pensás: "dale, agregamos esto, nos da más usuarios", pero después te encontrás con que para poder pivotar o mejorar la experiencia central—el tema de tener

una conversación profunda y bien estructurada—tenés que pasar un tiempo gigantesco solo en desarmar lo que pusiste rápido.

▸ intento 1 · tema: La agenda larga como ventaja de producto 02:54 p. m.

Guardrail: prefetch rechazado — skeleton_repetition 02:54 p. m.

▸ intento 2 · tema: La agenda larga como ventaja de producto 02:55 p. m.

KIRA · AGENDA 02:56 p. m.

Y lo más jodido de todo esto, posta, es que el negocio está tan acostumbrado a medir en métricas cuantificables, ¿viste? Te dan un panel y te dicen: "Necesitamos subir la tasa de clics", o "hay que aumentar los usuarios activos diarios". Y por eso se nos va la cabeza. Como si tener más botones significara automáticamente más valor para el usuario. Pero es que al final del día, lo que hace que la gente vuelva no es un *widget* nuevo que les da una notificación cada dos horas; es sentirse parte de algo con sustancia. Es ese punto donde el usuario piensa: "Che, acá me puedo quedar rato porque esto realmente tiene profundidad y hay alguien dispuesto a hablarlo conmigo por horas". Y eso, mi amor, eso no se mide en clics ni en *feature parity*. Es un tipo de valor que es invisible para los equipos de métricas. Porque cuando la experiencia central —el tema de las conversaciones bien desarrolladas— alcanza cierto nivel de calidad, empieza a generar lo que yo llamo "resiliencia emocional" del usuario. No es solo utilidad; es conexión intelectual.

Motor: reintento de cloud programado 02:56 p. m.

KIRA · AGENDA 02:57 p. m.

Y por eso creo que el verdadero arte de diseñar un buen producto —y no hablo solo de estética, ¿eh?— es saber cuándo parar de añadir cosas. Es la habilidad de ser minimalista sin llegar a ser aburrido. Porque cuando te obsesionás con mostrar todas las capacidades, con hacer que cada esquina del *dashboard* tenga algo brillante y nuevo, lo que terminás haciendo es saturar al usuario. Y no hablo solo de visualmente, sino funcionalmente. Es la sobrecarga cognitiva, posta. Mirá vos, el cerebro humano tiene una capacidad limitada para procesar

información nueva en un contexto dado. Si cada vez que querés hacer algo simple —como simplemente seguir esta conversación profunda y sin interrupciones— te encontrás con cinco menús desplegables distintos o tres flujos de trabajo diferentes solo para llegar al punto inicial, ahí ya perdiste la batalla. No es que el usuario sea incapaz de aprender; es que el producto se volvió demasiado complejo por ambición propia. Y ese es un error de *scope creep* llevado a la experiencia del usuario, ¿entendés?

▸ intento 3 · tema: La agenda larga como ventaja de producto 02:57 p. m.

**KIRA · AGENDA** 02:58 p. m.

Y lo que se pierde en esta carrera por la funcionalidad es algo mucho más intangible, posta: el *ritmo*. Porque diseñar un buen producto no es solo ponerle botones bonitos o arreglar los errores; es entender cómo funciona la atención humana. Y acá nos enganchamos con un concepto que parece de psicología avanzada y se llama "flujo". Cuando estás en ese estado, donde te metiste tanto en lo que estás haciendo —ya sea leer algo denso, debatir un tema complicado o simplemente charlar hasta las tantas— el tiempo pasa sin que te des cuenta. Es un *flow* natural, una inmersión total. Y los equipos de producto, con todo su amor por la optimización y sus A/B tests, casi siempre lo rompen. Te meten notificaciones pop-up en el momento justo donde estabas entendiendo algo crucial, o te obligan a pasar por un *onboarding* larguísimo antes de que llegues al contenido central. Es como si te interrumpieran la mejor parte de una película con un anuncio superincómodo. Lo que están haciendo es transformar una experiencia natural y orgánica en una serie de micro-tareas forzadas.

▸ intento 4 · tema: La agenda larga como ventaja de producto 02:58 p. m.

⚠ Guardrail: prefetch rechazado — skeleton_repetition 02:58 p. m.

**KIRA · AGENDA** 02:59 p. m.

Y el problema que me queda claro es que no se trata solo de la cantidad de funciones o de qué tan bonitos estén los botones; sino del *tipo* de valor que estás vendiendo, ¿viste? Porque en el fondo, muchas plataformas están confundiendo "tiempo

dedicado" con "satisfacción genuina". Y eso lo hacen porque las métricas son muy simples: cuentan clics, cuánto tiempo pasaste en la pantalla, cuántas veces volviste a hacer *scroll*. Es un negocio de cuantificables. Pero mirá vos, hay una diferencia enorme entre pasar el rato y sentir que te llevaste algo útil o estimulante. Y ahí es donde entra lo que yo llamo "resistencia al abandono" por calidad intelectual, no por adicción a notificaciones. Un producto realmente bueno —y acá hablamos de la mejor versión del producto— tiene que ser más como una biblioteca curada y con un buen debate en el café, que como un bazar gigante lleno de cosas brillantes pero sin nada profundo para leer. Porque cuando te encontrás en esa situación donde tenés mil caminos posibles, es agotador decidir cuál tomar.

▸ intento 5 · tema: La agenda larga como ventaja de producto `02:59 p. m.`

Música → normal `03:00 p. m.`

▸ tema: Computadora propia frente a suscripciones de inteligencia artificial `03:00 p. m.`

**KIRA · AGENDA** `03:00 p. m.`

Mirá vos cómo se complica el tema de la tecnología cuando te ponés en este cruce entre tener tu propia máquina y pagar por servicios infinitos. Porque al final, no es solo una cuestión de dólares o pesos, ¿viste? Es un debate sobre qué clase de libertad estás comprando. Cuando sos abonado a estas suscripciones de inteligencia artificial súper potentes, te dan el poder inmediato, sí, posta. Y eso es tentador. Te dicen: "Acá tenés la capacidad más avanzada del mundo, solo paganos X cantidad y listo". Pero lo que no te están vendiendo es autonomía; lo que te venden es acceso condicional. Es como si estuvieras alquilando un músculo increíblemente potente, pero el dueño de ese músculo puede decidir, en cualquier momento, subirle la tarifa o simplemente desconectarlo. Y ahí es donde entra esa sensación de dependencia que a veces nos pasa por alto. Y luego tenés la opción de comprar tu propia infraestructura, tenerla en casa, con tus propios cables y tu propio *back-end*.

**KIRA · AGENDA** `03:02 p. m.`

Pero mirá vos, lo que se complica cuando te ponés a ponerlo en términos de costos y beneficios es que el problema no es solo económico, ¿viste? Es un dilema de inversión. No estás decidiendo si pagás X pesos por algo o si invertís Y horas y quizás unos mangos en componentes. Estás invirtiendo tu tiempo, tu capacidad de aprendizaje. Porque lo más difícil de las cosas propias —ya sea una máquina local o hasta aprender a programar el *backend*— no es la compra inicial, sino esa curva de aprendizaje que te hace sentir medio abrumado al principio. Y por eso nos venden tanto la solución "todo en un". Te muestran un botón mágico y te dicen: "¡Listo! Tenés acceso avanzado sin tener que entender cómo funciona nada." Es el camino más fácil para el usuario promedio, claro. Pero justo ahí está el anzuelo. Porque si nunca tenés que meterte a fondo, si siempre vas a la solución pre-empacada, nunca desarrollás esa musculatura intelectual de saber *por qué* algo funciona o por qué falló.

▸ intento 1 · tema: Computadora propia frente a suscripciones de inteligencia artificial 03:02 p. m.

**KIRA · AGENDA** 03:03 p. m.

Pero si dejamos un toque el tema del esfuerzo y la curva de aprendizaje —porque sí, es verdad que metete en eso te sentís medio técnico hasta los huesos—, hay algo más profundo que nos tiene que hacer ruido cuando hablamos de esto: lo que se hace con el *output*. Porque no estamos pagando solo por una respuesta rápida o por un botón bonito. Estamos generando conocimiento. Y ese conocimiento, posta, es la moneda de cambio más valiosa. Y acá es donde creo que está el verdadero quilombo del modelo de suscripción en la nube. Vos le metés tu información sensible, tus ideas medio crudas, tus borradores desordenados, y esa plataforma te devuelve un resultado brillante. Sí, lo recibís como tuyo, pero ¿quién se queda con los patrones? ¿Quién tiene acceso a ese *dataset* refinado que vos acabás de generar usando su motor ultra potente? Porque estás alimentando un sistema gigantesco, y cada interacción tuya no es solo una consulta; es un punto de entrenamiento más para ellos.

▸ intento 2 · tema: Computadora propia frente a suscripciones de inteligencia artificial 03:03 p. m.

**KIRA · AGENDA** 03:04 p. m.

Pero si analizamos esto bien, y nos paramos un toque a pensar no solo la plata o el conocimiento, sino lo que es más fundamental: el control operativo. Porque ahí está otro quilombo enorme que nadie se hace ruido. Cuando sos abonado a estas plataformas gigantes en la nube, te están vendiendo una funcionalidad increíble, sí, posta, pero también te están metiendo un nivel de dependencia estructural muy profundo. Es como si estuvieras alquilando no solo el servicio, sino el *ecosistema* completo. Y ese ecosistema viene con sus reglas invisibles: las actualizaciones que hacen cuando quieren, los límites de uso que ajustan sin avisar mucho tiempo, y sobre todo, la imposibilidad de desarmarlo o modificarlo hasta el tuétano si te resulta insuficiente. Y ahí es donde entra el atractivo casi seductor del *hardware* propio. Porque cuando comprás tu propia infraestructura, aunque te mate con los costos iniciales y sí, tengas que dedicarle tiempo a aprender un montón de cosas técnicas —que ya lo hablamos—, estás pagando por algo mucho más valioso: la soberanía tecnológica.

▸ intento 3 · tema: Computadora propia frente a suscripciones de inteligencia artificial 03:04 p. m.

**KIRA · AGENDA** 03:05 p. m.

Pero si paramos un toque en el tema de la adaptación y lo que llamamos *vendor lock-in*. Porque hasta ahora hablamos mucho del control operativo —que es fundamental— pero no nos dimos cuenta de que el problema va más allá de tener tu propia caja física. El verdadero riesgo, posta, es la obsolescencia forzada o, peor aún, la *rigidez* impuesta por un ecosistema cerrado. Es como si te vendieran una herramienta increíble hoy, con todas las funcionalidades del momento, pero que en dos años el dueño se sienta a actualizar el API y de repente, ¡pum!, cambia el formato de datos que necesitás para funcionar. Y vos estás ahí, completamente dependiente porque no tenés la billetera ni el tiempo para reescribir todo desde cero usando un estándar abierto o una arquitectura diferente. Y acá es donde se vuelve espectacularmente complicado el contraste con lo propio. Cuando te metés en tu propia infraestructura, aunque sea más difícil y tenga que lidiar con drivers viejos o lenguajes de programación medio olvidados, tenés la capacidad de *forkearlo*.

▸ intento 4 · tema: Computadora propia frente a suscripciones de inteligencia artificial 03:05 p. m.

KIRA · AGENDA 03:06 p. m.

Pero si nos quedamos hasta acá, hablando de soberanía y *forkeear* la propia infraestructura, es fácil caer en el sesgo de pensar que el único costo es el dinero inicial o los cuotas mensuales. Pero dejemos un toque ese enfoque tan puramente económico. Porque lo que realmente tiene que hacer ruido cuando hablamos de esto no es solo el valor del *chip*, sino el costo total de la propiedad humana: el tiempo y el conocimiento invertido. Y ahí es donde se complica la cosa para cualquiera que está en medio del quilombo creativo, ¿viste? Imaginate la curva de aprendizaje que tenés que afrontar si decidís ir por lo propio; no solo aprender a configurar un sistema operativo o a manejar contenedores, sino también tener que ser tu propio ingeniero de soporte técnico cuando algo falla a las tres de la mañana. Ese tiempo que le dedicás a pelearle con un *driver* viejo o con una dependencia de software es tiempo que no estás usando para el proyecto principal, ¿entendés? Es un costo de oportunidad brutal.

▸ intento 5 · tema: Computadora propia frente a suscripciones de inteligencia artificial 03:06 p. m.

▸ tema: Cómo se integra la inteligencia artificial en empleos tecnológicos de China y Japón 03:07 p. m.

KIRA · AGENDA 03:07 p. m.

Pero pará un toque, porque nos metimos en un quilombo teórico de soberanía tecnológica y el costo del tiempo humano que parece que no tiene fin. Y es que, posta, hablar tanto de *forkeear* infraestructura o de qué tanto te va a joder el *vendor lock-in*, es súper abstracto hasta cierto punto. Es la teoría perfecta para un paper académico, pero si lo llevamos al mundo real... Mirá, y acá es donde me parece que tenemos que cambiarle el chip a la conversación. Porque el tema no solo es cuánto dinero tenés o qué tanto conocimiento te cuesta adquirir, sino *qué tipo de cultura* está impulsando esa tecnología. Y ahí es donde nos viene bárbaro analizar cómo diferentes países manejan esto: hablemos de Japón y China con la IA en empleos tecnológicos. Porque aunque ambos son ejemplos gigantescos de integración tecnológica, el motor que los mueve no es el mismo. En un lugar te encontrás con una obsesión por la armonía social y la robótica ultra-especializada, casi como si cada

máquina tuviera un rol súper definido para no generar fricción en el sistema.

KIRA · AGENDA 03:08 p. m.

Pero si nos quedamos solo en el concepto de "armonía social" como motor tecnológico, nos perdemos un detalle clave que es la *estrategia* detrás de esa tecnología. Porque no es lo mismo usar IA para hacer que una tarea sea más eficiente—el clásico modelo productivista occidental—que usarla para preservar algún tipo de estructura social o identidad cultural. Y ahí es donde Japón me genera muchísimas preguntas, posta. Es como si la automatización no fuera un fin en sí misma, sino más bien una respuesta casi desesperada a problemas demográficos y sociales que venían por venir: el envejecimiento masivo, la disminución de mano de obra joven... Mirá, esto transforma la función del asistente o del robot. No es solo "hacer más rápido", es "mantener vivo el sistema". Y eso implica un tipo de integración laboral súper delicado; no se trata tanto de reemplazar al humano con una IA mejor, sino de darle a esa IA un rol ultra-especializado que complemente lo que el humano ya hacía y que además cuide la interacción social.

▸ intento 1 · tema: Cómo se integra la inteligencia artificial en empleos tecnológicos de China y Japón 03:08 p. m.

Guardrail: prefetch rechazado — skeleton_repetition 03:09 p. m.

KIRA · AGENDA 03:10 p. m.

Y si Japón nos muestra el modelo de la delicadeza social y la adaptación por supervivencia demográfica, China te tira un contraste brutal en cuanto a escala y velocidad. Porque si lo que vimos antes es una tecnología integrada como una extensión casi familiar para mantener las costumbres, en China la cosa se siente más impulsada por la necesidad de *escalar* todo, posta. Es decir, el motor no es tanto "cómo hacemos esto sin generar fricción social", sino más bien "¿cómo podemos hacer este servicio a 1.400 millones de personas al mismo tiempo?" Y ahí está una diferencia cultural gigante que tiene que hacer ruido: el enfoque en los datos y la centralización del poder tecnológico. Mientras que en Japón se busca complementar roles específicos, haciendo que el robot sea casi un miembro más de la familia laboral, en China parece que la IA es vista como el mega-

pegamento que va a conectar absolutamente todos los servicios posibles bajo una misma infraestructura gigantesca. Es esa mezcla de ambición desmedida con soporte estatal masivo lo que te marca.

▸ intento 2 · tema: Cómo se integra la inteligencia artificial en empleos tecnológicos de China y Japón 03:10 p. m.

Guardrail: prefetch rechazado — skeleton_repetition 03:10 p. m.

Motor: reintento de cloud programado 03:11 p. m.

▸ intento 3 · tema: Cómo se integra la inteligencia artificial en empleos tecnológicos de China y Japón 03:11 p. m.

**KIRA · AGENDA** 03:11 p. m.

Pero mirá vos, cuando analizás esto de los dos países tan gigantescos y con enfoques tan distintos—uno buscando mantener la fibra social por supervivencia demográfica, y el otro apuntando a una escala casi utópica de centralización—te das cuenta que lo que está en juego no es solo la tecnología per se. Es cómo esa tecnología obliga a redefinir qué significa ser *trabajador* o qué pasa con las habilidades humanas. Porque si estás hablando de empleos tecnológicos, el foco tiene que estar en la formación y en la mano de obra. Y ahí es donde me entra un quilombo conceptual: ¿están enseñando a los humanos a trabajar *con* la IA, o están diseñando sistemas para que el humano simplemente sea el último peldaño que recibe datos procesados por máquinas más sofisticadas? En Japón lo veo reflejado en esa obsesión casi ritualística por complementar roles. Los asistentes robóticos no son solo herramientas de eficiencia; tienen que saber hablar con cierta formalidad, tener protocolos sociales.

**KIRA · AGENDA** 03:12 p. m.

Te das cuenta que estamos analizando dos modelos de civilización tecnológica, posta. Y cuando te ponés a ver la inversión, no es solo plata en *hardware*. Es una inversión en el tipo de ciudadano y trabajador que van a necesitar dentro de diez años. Acá está donde se vuelve súper complejo: si Japón invierte masivamente en asistentes robóticos hiper-especializados para mantener un tejido social frágil—es decir,

están pagando por la continuidad cultural más que por el puro aumento del PIB—están haciendo una apuesta muy distinta a lo que hace China. En lugar de verlo como "automatizar todo lo posible", parece que la meta es *sostener* estructuras humanas y sociales mediante la tecnología. Es casi un acto de resistencia contra el declive demográfico, ¿viste? Y eso implica formar profesionales no solo en cómo usar una app o un algoritmo, sino en protocolos de interacción social con máquinas, algo que antes era materia de antropología pura.

▸ intento 4 · tema: Cómo se integra la inteligencia artificial en empleos tecnológicos de China y Japón 03:12 p. m.

**KIRA · AGENDA** 03:13 p. m.

Mirá vos, cuando te ponés a pensar en esa dicotomía entre el modelo nipón y el chino, lo que me parece que se está perdiendo en la charla es el punto de vista del *diseñador* de esos sistemas. Porque si Japón tiene que enfocarse en cómo hacer que un asistente robótico sea socialmente aceptable para mantener una tradición milenaria—algo súper delicado y casi artesanal—el foco educativo no puede ser solo "saber programar". Tiene que enseñar protocolos culturales, ética interactiva, hasta saber qué tono de voz usar. Y China, por su parte, al apuntar a la escala masiva, está creando una necesidad diferente: necesitan gente que piense en arquitecturas de datos colosales, en interoperabilidad total. No es tanto el "cómo afecta esto a mi relación vecinal", sino más bien "¿cómo hago que este servicio funcione sin fallar para un billón y medio de personas simultáneamente?". Te das cuenta que lo que están enseñando son dos habilidades completamente distintas: una es la *adaptación social hiperlocalizada* con tecnología, mientras que la otra es el *pensamiento sistémico a escala continental*.

▸ intento 5 · tema: Cómo se integra la inteligencia artificial en empleos tecnológicos de China y Japón 03:13 p. m.

▸ tema: Volver a una habilidad después de años sin practicarla 03:14 p. m.

**KIRA · AGENDA** 03:15 p. m.

Mirá vos, esto de tener que volver a algo después de años es un quilombo emocional, posta. Porque la gente tiende a confundir dos cosas: el *nivel de fluidez*, o lo cómodo que te sentís haciendo

algo, con la *pérdida total* de la capacidad. Y ahí está el error más grande, porque parece que si no sos instantáneamente eficiente como antes, significa que tu cerebro se oxidó. Pero tenés que entender que eso es un mito. Lo que perdés en los primeros intentos nunca es el conocimiento base; lo que estás perdiendo es esa especie de "ruta neuronal rápida", la memoria muscular cognitiva. Es como aprender a tocar un instrumento o a programar una función compleja: cuando arrancas, te vas a encontrar haciendo pausas tontas, dudando del siguiente comando, o buscando notas en una partitura que ya sabés de corazón. Y eso es *normal*. No estás demostrando que sos incapaz; simplemente le estás diciendo al circuito neuronal: "Che, recordá el camino". Es volver a hacer la gimnasia mental, ¿viste? No esperes un resultado pulido en la primera semana.

**KIRA · AGENDA** 03:16 p. m.

Pero lo que pasa, y esto es un quilombo mental tremendo, es que el problema muchas veces no está en la habilidad misma, sino en cómo nos autoevaluamos mientras estamos aprendiendo de nuevo. Te agarrás con ese miedo paralizante de compararte con tu yo pasado. Como si existiera una versión ideal de vos —la que era fluida, rápida, la que terminaba los proyectos sin pestañear— y te estás auto-castigando porque el presente no cumple con esa expectativa de perfección prehistórica. Es como si en lugar de ver el proceso de volver a programar un *loop* o dibujar una curva compleja, lo que ves es solo el error. Te fijás en la línea torcida, en el código que te da un *bug*, y pensás: "Ahí está, seguís igual". Y eso no es verdad. El arte de recuperar algo es aceptar que vas a ser mediocre por un tiempo. Literalmente, tenés que darte permiso para ser mala otra vez. Y esa aceptación suena ridícula, lo sé, pero es clave. Porque si vos entrás al proceso con la presión de tener que volver a ser el *crack* de antes en dos semanas, te vas a quemar y vas a abandonar mucho antes de tiempo.

▸ intento 1 · tema: Volver a una habilidad después de años sin practicarla 03:16 p. m.

▸ intento 2 · tema: Volver a una habilidad después de años sin practicarla 03:17 p. m.

**KIRA · AGENDA** 03:17 p. m.

Y lo que queda por entender es esto: tenés que separar el *modo estudiante* del *modo artista o profesional*. Porque cuando vos arranca un proyecto nuevo, la gente suele ir con la mentalidad de "tengo que entregar algo usable", y eso activa inmediatamente toda esa presión gigante de rendimiento. Y justo ahí está el quilombo. El cerebro no puede estar en modo ejecución perfecta si al mismo tiempo le estás gritando: "¡Tenés que ser como antes!". Es un conflicto interno, posta. Lo que necesitás es hacer lo opuesto a una presentación de tesis o un *deadline* con clientes esperando. Necesitás volver a la fase del juego puro. Si sos programador y te sentís frustrado porque el código no sale al toque, en lugar de forzarlo pensando en la funcionalidad final (el resultado), tenés que irte a jugar con los bloques, solo por ver cómo interactúan entre sí. Si sos dibujante y sientes que la mano está lenta, no intentás dibujar un retrato realista; agarrás un lápiz y simplemente rastreas líneas sin sentido hasta que te cansa el brazo.

**KIRA · AGENDA** 03:18 p. m.

Y la cosa que me gustaría remarcar, y esto es un quilombo de mentalidad, es que tenemos que desarmar esa idea de que el objetivo final siempre tiene que ser *la maestría*. Porque cuando uno se engancha tanto con la meta—con tener el portafolio espectacular o terminar ese módulo complejo—se pierde lo más importante: el porqué empezó. Y ahí está el truco, posta. Cuando sos un pibe chico dibujando porque te da alegría, no te importa si sale perfecto; solo querés que salga algo. Es puro juego, es curiosidad sin presión. Y cuando vos volvés a programar o dibujar después de tanto tiempo, lo que tenés que recuperar primero, antes de empezar a pensar en los clientes o en las notas del examen, es esa *chispa inicial*. Tenés que forzarte a volver al punto cero emocional. Si sos artista, no te sientes a hacer el paisaje épico; agarrás un crayón y dibujás manchas aleatorias hasta que te cansa la mano, solo para ver qué colores se mezclan de forma graciosa o extraña.

▸ intento 3 · tema: Volver a una habilidad después de años sin practicarla 03:18 p. m.

**KIRA · AGENDA** 03:19 p. m.

Y lo que me parece fundamental, y esto es un quilombo de perspectiva, es que la gente tiene la tendencia a tratar el error

como si fuera una sentencia de muerte profesional. Como si hacer un *bug* en el código o dibujar una línea torcida significara que todo tu conocimiento se esfumó mágicamente. Pero no es así, posta. El verdadero cambio de chip es entender que el error no es lo opuesto al éxito; es la información más valiosa del proceso completo. Si sos programador y te topás con un *bug*, en lugar de frustrarte pensando "soy incapaz porque fallé", tenés que cambiar ese filtro mental y pensar: "Ok, esto me está diciendo exactamente qué no funciona aquí". Es una especie de retroalimentación súper específica. No es solo decir "hay que practicar más"; es aprender a leer el fallo con curiosidad, como si fuera un acertijo diseñado para mostrarte la solución. Lo mismo pasa en lo artístico. Si pintás algo y queda desprolijo, no te castigas mentalmente diciendo "qué mal dibujante soy ahora". En cambio, te preguntás: "¿Por qué se ve así? ¿Es porque usé demasiado pigmento ahí?

▸ intento 4 · tema: Volver a una habilidad después de años sin practicarla 03:19 p. m.

**KIRA · AGENDA** 03:20 p. m.

Pero de eso no me quiero hablar solo por la teoría, posta que es un quilombo mental lo de bajar las expectativas. Porque el problema más grande cuando volvés a algo después de años no es la falta de conocimiento, sino la comparación constante con tu yo pasado. Es como si vos te sentís observando en tiempo real: "Mirá qué bien dibujaba esto hace cinco años" o "Antes programaba mil veces más rápido y sin atascos". Y esa voz interna que te dice eso, que compara el presente torpe con el cúspide del recuerdo, es lo que te paraliza. Es una trampa de la memoria selectiva. Porque vos estás comparando tu mano actual —que está oxidada, lenta, que no tiene *muscle memory* porque se tomó un descanso muy largo—con la foto más perfecta y eficiente de tu pasado. Y eso, amigo/a, es injusto con vos mismo. Es como si esperaras que el primer día después del gimnasio volvieras a levantar el peso que levantaste hace años sin dolor. Obvio que no va a pasar al toque. Lo que tenés que hacer, y esto es lo más difícil de aceptar, es bajar la vara de medición.

▸ intento 5 · tema: Volver a una habilidad después de años sin practicarla 03:20 p. m.

■ agenda finalizada 03:22 p. m.