PER A TOTA LA TRIPULACIO

La Tripulacio Agentica

Guia del Tripulant

Rasmus Bornhoft Schlunsen

Marc 2026

rev 63

Una nota abans de comecar

Aquest llibre es un company de *La Tripulacio Agentica: Enginyeria en l'era dels agents d'IA*. Aquell llibre va ser escrit per a enginyers de programari – persones que escriuen codi per guanyar-se la vida. Aquest es per a tots els altres que se'n surten be amb els ordinadors.

Potser ets un gestor de projectes, un dissenyador, un administrador de sistemes, un analista de dades, un petit empresari, o simplement la persona de la teva familia que arregla el Wi-Fi. Vius entre fulls de calcul, gestiones bústies de correu com un general i ara mateix tens quaranta-set pestanyes del navegador obertes. Pero no has escrit mai una linia de codi – i no cal que ho facis.

Els agents d'IA estan canviant com es construeix el programari. Aixo t'afecta, tant si treballes al costat de desenvolupadors, com si gestiones un negoci que depen del programari, o simplement intentes entendre que passa al mon que t'envolta. Les idees d'aquest llibre son reals. Hem simplificat els detalls tecnics, pero no els hem aigualit.

Quan acabis, entendras que son realment els agents, com funciona el programari modern per dins i – el mes important – com dirigir un agent perque construeixi alguna cosa real. No cal saber programar.

Rasmus Bornhoft Schlunsen
Marc 2026

Que hi ha dins

Benvingut a la tripulacio

Figura 1: *L'horitzó és més ampli del que penses.*

Fa sis mesos, el meu amic Morten em va dir que volia construir una aplicacio per al seu negoci. En Morten instal · la finestres – les de vidre, no les de Microsoft. Fa quinze anys que s'hi dedica. Coneix cada model de finestra, cada tipus de marc, cada truc per aconseguir un segellat perfecte en una vella masia danesa. Dirigeix un equip de quatre homes, gestiona pressupostos en paper i fa el seguiment dels treballs en un full de calcul que faria plorar un comptable.

En Morten es brillant amb els ordinadors. Es va muntar el seu propi NAS, te un servidor Plex, va automatitzar la meitat de la seva llar intel · ligent amb scripts que va trobar a Reddit. Pero no ha escrit mai una linia de codi.

Li vaig parlar dels agents d'IA – eines que poden escriure codi, construir interficies, configurar bases de dades, tot a partir d'instruccions en llenguatge natural. Li vaig ensenyar el basic. Com descriure el que vols. Com dividir una gran idea en petites tasques. Com revisar el resultat i corregir el rumb quan les coses se'n van de mare.

El que va passar despres ens va sorprendre a tots dos. En un cap de setmana, en Morten tenia un prototip funcional. Una eina de pressupostos on podia introduir les mides de la paret, triar un model de finestra, veure una previsualitzacio en 3D de com quedaria i generar un pressupost en PDF per enviar per correu al client. No era perfecte – la previsualitzacio 3D tenia un problema d'il·luminacio i el format del PDF necessitava feina. Pero era *real*. Funcionava. I ho va construir sense escriure ni una sola linia de codi.

Els seus clients van comecar a comentar com de professionals quedaven els seus pressupostos. El seu equip va comecar a utilitzar-ho en tauletes a les obres. Un competidor li va preguntar qui li havia fet el programari.

En Morten te zero anys d'experiencia en programacio. Simplement va aprendre a dirigir alguna cosa que si que en sap.

Que es aquest llibre

Aquesta es la guia que m'hauria agradat poder donar a en Morten abans que comences. Esta construida sobre els mateixos principis que l'edicio d'enginyeria de *La Tripulacio Agentica*, pero traduida per a persones que treballen *amb* la tecnologia sense construir-la des de zero.

Aprendras que son realment els agents d'IA – despullats del llenguatge de marketing. Aprendras com funciona el programari modern per dins, perque entenguis que estan construint els agents quan construeixen. Aprendras a comunicar-te amb els agents amb prou claredat perque facin el que vols dir, no el que has dit. I aprendras quan confiar en el resultat, quan objectar i quan fer la feina tu mateix.

Aixo no es un llibre sobre prompts i trucs. Es un llibre sobre *pensar* – el tipus de pensament que converteix una bona idea en un producte funcional, tant si ets tu qui escriu el codi com si no.

Per a qui es

Ets bo amb els ordinadors. Potser molt bo. Gestiones sistemes, domines dades, configures eines i resols problemes que enviarien la majoria de la gent al servei d'assistencia tecnica. Ets la persona a qui truquen els teus amics quan la impressora no es connecta.

Pero no has obert mai un editor de codi i has escrit `function`. No has desplegat mai un servidor web. No t'has quedat mai mirant una terminal plena de missatges d'error i has sabut, instintivament, quina linia arreglar.

No passa res. No cal que ho facis.

El que necessites es un model mental – una manera d'entendre que son aquestes eines, que poden fer i com dirigir-les. Perque la persona que sap *que* construir i ho pot descriure clarament es, cada vegada mes, la persona mes important de la sala. Mes important que la persona que pot teclejar el codi. La part del codi s'abarateix dia a dia. La part del pensament no ho fara mai.

Com llegir aquest llibre

Aquest llibre es un viatge, no un manual de referencia.

Comencem amb el que esta canviant al mon i per que et concerneix. Despres obrim el capo del programari modern – no per convertir-te en enginyer, sino per donar-te el vocabulari i el model mental per entendre amb que treballen els agents. A partir d'aqui, entrem en els agents mateixos: que son, com parlar-hi, com posar-hi limits i com verificar la seva feina.

A la segona meitat, construim alguna cosa real. No un exemple de joguina. Una aplicacio funcional per a un negoci real, dirigida completament a traves de conversa amb un agent. I tanquem amb les preguntes mes dificils – quan les coses van malament, quan intervenir i quin es el teu paper en un mon on el codi s'escriu sol.

Quan acabis, no seras programador. Pero seras alguna cosa que potser importa mes: algu que en pot dirigir un.

El tema nautic

Potser notes que aquest llibre parla molt de vaixells, tripulacions i capitans. No es casualitat. L'edicio principal de *La Tripulacio Agentica* utilitza la metafora d'una tripulacio de vaixell al llarg de tot el text – l'enginyer com a capita, els agents com a membres de la tripulacio. L'edicio infantil ho porta encara mes lluny, amb personatges il · lustrats i aventures al mar.

En aquesta edicio, tu ets el tripulant. No el capita – aquest es l'enginyer que construeix el vaixell. No un passatger – no estas aqui nomes per fer el viatge. Ets una part vital de la tripulacio. Saps coses que el capita no sap. Veus coses des de coberta que no

son visibles des del timo. I cada vegada mes, ets tu qui diu a la tripulacio cap a on navegar.

Tot gran vaixell necessita un capita. Pero cap capita ha navegat mai sol.

El terra s'esta movent

Figura 2: *La línia entre tècnic i no tècnic s'està dissolent.*

El mes passat, una amiga meva – gestora de projectes en una agencia mitjana – havia de preparar un informe d'analisi competitiva. Normalment hi hauria dedicat dos dies: recopilant dades, creuant preus, construint taules comparatives, escrivint el resum. Es el tipus de treball de coneixement que requereix judici pero que es sobretot rutinari.

Va obrir un agent d'IA, va descriure el que necessitava, li va donar acces als documents rellevants i algunes fonts web, i li va demanar que redactes l'informe. Quaranta minuts despres, tenia un primer esborrany que cobria el 80% del que ella hauria produit en dos dies. Va dedicar una hora mes a refinar-lo – corregint algunes

inexactituds, ajustant el to per a la seva audiencia, afegint context que nomes ella coneixia. Acabat abans de dinar.

No es va convertir en programadora. Es va convertir en alguna cosa potser mes important: algu que sap com dirigir una eina potent cap a un objectiu clar.

Aixo esta passant a tot arreu. No nomes a empreses de programari. No nomes amb desenvolupadors. Persones que mai s'havien imaginat treballar amb IA estan descobrint que la linia entre "tecnic" i "no tecnic" s'esta dissolent. No perque tothom estigui aprenent a programar – sino perque les eines estan aprenent a escoltar.

La vella frontera

Durant decades, hi havia una frontera clara. A un costat: persones que sabien programar. Construien el programari, els webs, les eines. A l'altre costat: tots els altres. Utilitzaves el programari, enviaves peticions de funcionalitats i esperaves.

Si tenies una idea per a una aplicacio, tenies tres opcions:

1. Aprendre a programar tu mateix – un viatge de diversos anys sense cap garantia d'arribar a ser prou bo.
2. Contractar un desenvolupador – car, lent, i et passaves la major part del temps intentant explicar el que volies.
3. Utilitzar una eina no-code – limitada, frustrant, i topaves amb un mur en el moment que volies qualsevol cosa personalitzada.

Les tres opcions tenien el mateix problema de fons: hi havia un *buit de traduccio* entre la teva idea i el programari funcional. Sabies el que volies. Simplement no podies dir-ho directament a un ordinador.

Aquest buit s'esta tancant.

El nou panorama

Els agents d'IA no eliminen la necessitat d'habilitats tecniques. Pero redueixen drasticament la barrera. La persona que abans necessitava un desenvolupador per construir una eina personalitzada ara pot descriure el que vol i fer que un agent ho produeixi. No perfectament. No sense iteracio. Pero *del tot* – i aixo es la revolucio.

Pensa que significa:

- La directora de vendes que necessita un CRM que funcioni com el *seu* equip ven realment – no com Salesforce creu que ven tothom.
- El propietari d'un restaurant que vol un sistema de comandes que gestioni la seva estructura de menu peculiar i les zones de repartiment locals.
- L'instal · lador de finestres que vol una eina de pressupostos amb previsualitzacions 3D.

Aquestes persones sempre han tingut les idees. Sempre han tingut l'expertesa de domini – el coneixement profund del seu propi camp que cap desenvolupador de programari posseeix. El que no tenien era la capacitat d'*expressar* aquest coneixement en un llenguatge que els ordinadors entenguessin.

Els agents s'estan convertint en aquest traductor.

Que no ha canviat

Abans que t'emocionis massa – i abans que els enginyers que miren per sobre la teva espatlla es posin massa nerviosos – permeteu-me que sigui clar sobre el que *no* ha canviat.

Construir bon programari segueix sent dificil. Un agent pot produir codi, pero produir codi no es el mateix que construir un sistema fiable. Hi ha una rao per la qual l'edicio d'enginyeria d'aquest llibre te capitols sobre testing, sandboxing, control de versions i pipelines de desplegament. L'agent et dona velocitat. No et dona judici.

Que significa aixo per a tu: les coses que tu aportes – expertesa de domini, empatia amb l'usuari, logica de negoci, gust – ara son mes valuoses que mai, no menys. El coll d'ampolla ja no es "podem construir-ho?" El coll d'ampolla es "sabem que construir i sabrem si es correcte?" Aixo es el teu departament.

Per que necessites aquest llibre

Perque ningu t'esta explicant aquesta transicio des del teu costat de la tanca.

Hi ha centenars de llibres per a enginyers sobre treballar amb IA. Hi ha cursos per a principiants que prometen ensenyar-te Python en trenta dies. Hi ha publicacions excitades sobre com la IA substituira a tothom.

Res d'aixo et serveix. No necessites convertir-te en enginyer. No necessites aprendre Python. I no t'estan substituint – si de cas, el teu paper s'esta expandint.

El que necessites es *prou* comprensio tecnica per treballar al costat d'aquestes eines i les persones que les construeixen. No per fer la

seva feina. Per fer la teva – millor, mes rapid i amb un lloc a la taula quan es prenen les decisions.

Aixo es el que ofereix aquest llibre.

Que hi ha sota el capó

Figura 3: *Cada aplicació és un restaurant — menjador, cuina i rebost.*

Un amic et diu que està construint una app. "És un backend amb Django, un frontend amb React, Postgres per a la base de dades i Redis per a la cache." Tu assents. Somrius. No tens absolutament ni idea del que vol dir res d'això.

Aquest capítol ho soluciona.

No et convertirem en enginyer. Et donarem el model mental — el mapa del territori — perquè quan sentis aquestes paraules, sàpigues de quina part de la màquina estan parlant. I, el que és més important, quan dirigeixis un agent perquè construeixi alguna cosa, entendràs què està construint realment.

Farem servir un exemple al llarg de tot el capítol: una eina de gestió de projectes, com una versió simplificada de Trello. Un tauler amb columnes, targetes que pots arrossegar i persones assignades a tasques. Prou senzill per entendre'l, prou complex per ser real.

El restaurant

Cada aplicació web està construïda a partir d'uns quants blocs bàsics. Diferents apps utilitzen eines específiques diferents, però la *forma* sempre és la mateixa. La manera més fàcil d'entendre-ho és pensar en un restaurant.

Un restaurant té un menjador — la part que veuen els clients. Té una cuina — la part que fa la feina de debò. Té un rebost — on es guarden els ingredients. I té una zona de preparació — on es mantenen a mà els ingredients més usats perquè el cuiner no hagi d'anar al rebost cada trenta segons.

Una aplicació web té les mateixes quatre parts. Simplement tenen noms diferents.

React — El menjador

Què és: React és un *framework de frontend*. És la part de l'aplicació que s'executa al teu navegador web — tot el que veus, toques i amb què interactues.

Al nostre clon de Trello: El tauler amb les seves columnes. Les targetes que pots arrossegar de "Pendent" a "En curs". El botó que diu "Afegir tasca". El petit avatar que mostra qui està assignat. El menú desplegable quan cliques els tres punts. Tot això és React.

Per què importa: React no emmagatzema cap dada. No pren decisions sobre regles de negoci. És el menjador — bellament disposat, responent a cada interacció teva, però no cuina el menjar. Quan arrossegues una targeta a una nova columna, React fa que la targeta es mogui *immediatament* a la teva pantalla. Però entre bastidors, està enviant un missatge a la cuina dient "el client vol moure aquesta targeta". Si la cuina diu que no — potser no tens permís — React torna a posar la targeta al seu lloc.

Què sentiràs dir: "És una app de React" vol dir que la part amb què interactues està construïda amb aquesta eina. Hi ha alternatives — Vue, Svelte, Angular — però totes fan aproximadament la mateixa feina. React és la més comuna, com la majoria de restaurants tenen un menjador independentment de quines cadires hagin triat.

Django — La cuina

Què és: Django és un *framework de backend*. És el cervell de l'aplicació — la part que s'executa en un servidor, fa complir les regles, processa les peticions i decideix què passa.

Al nostre clon de Trello: Quan cliques "Afegir tasca" i escrius "Comprar queviures", React envia aquesta petició a Django. Django comprova: Estàs connectat? Pertanys a aquest projecte? El títol de la tasca és vàlid? El projecte encara és actiu? Si tot és correcte, Django diu a la base de dades que emmagatzemi la nova tasca i envia una confirmació a React.

Per què importa: Django és on viu la *lògica*. Les regles de preus. El sistema de permisos. La lògica de negoci que diu "només els administradors del projecte poden eliminar taulers". És la cuina — no li importa quin aspecte té el menjador, i al menjador no

li importa quina marca de forn utilitza la cuina. Es comuniquen mitjançant una nota de comanda estandarditzada, de la qual parlarem en un moment.

Què sentiràs dir: "El backend retorna un error" vol dir que Django (la cuina) s'està ofegant amb una petició. "Hem d'afegir aquesta lògica al backend" vol dir que la regla ha de viure a Django, no a React. Hi ha alternatives — Rails (Ruby), Express (JavaScript), Laravel (PHP), FastAPI (Python, com Django) — però el concepte és idèntic. Una cuina és una cuina.

Postgres — El rebost

Què és: Postgres (abreviatura de PostgreSQL) és una *base de dades*. És on viuen totes les dades — cada compte d'usuari, cada projecte, cada tasca, cada comentari, cada marca de temps.

Al nostre clon de Trello: Quan Django emmagatzema aquella nova tasca, va a parar a Postgres. Imagina't un arxivador enorme i meticulosament organitzat. Hi ha un calaix etiquetat "Tasques", i dins seu, cada tasca és una fila:

ID	Títol	Projecte	Estat	Creat
247	Arreglar la barra de navegació	Projecte 3	Fet	12 de març
248	Comprar queviures	Projecte 3	Pendent	18 de març
249	Trucar al client	Projecte 5	En curs	18 de març

Quan carregues el teu tauler, Django demana a Postgres: "Dóna'm totes les tasques del Projecte 3, ordenades per estat." Postgres les troba i les retorna. Django les passa a React. React dibuixa el tauler. Tot el viatge dura uns 200 mil · lisegons — menys temps que un parpelleig.

Per què importa: La base de dades és l'única part del sistema que *recorda* les coses. Si el servidor es reinicia, Django arrenca de nou des de zero — però totes les dades estan segures a Postgres, exactament on es van deixar. Si perds la base de dades, ho has perdut tot. Per això les còpies de seguretat importen.

Què sentiràs dir: "Està a la base de dades" vol dir que la informació està emmagatzemada permanentment. "Hem de consultar la base de dades" vol dir que hem de demanar a Postgres dades específiques. "La base de dades va lenta" vol dir que Postgres triga massa a trobar les coses — normalment perquè l'arxivador s'ha fet enorme i ningú ha construït un índex (pensa en un índex com les pestanyes als calaixos que et deixen saltar directament a la secció correcta).

Redis — La zona de preparació

Què és: Redis és una *cache en memòria*. Manté les dades consultades freqüentment en un emmagatzematge ràpid i temporal perquè l'aplicació no hagi de consultar la base de dades per cada petició.

Al nostre clon de Trello: Alguna informació es demana constantment. "Quantes notificacions no llegides té aquest usuari?" "Quantes tasques hi ha al Projecte 3?" En lloc de fer que Postgres remeni l'arxivador cada vegada que algú carrega la pàgina, Django emmagatzema la resposta a Redis.

Redis manté tot en memòria — pensa-hi com una pissarra muntada a la paret de la cuina. Necessites el recompte de notificacions? Mira la pissarra. Instantani. Quan el recompte canvia (algú t'assigna una nova tasca), Django actualitza la pissarra. Si la pissarra s'esborra — un reinici del servidor, per exemple — no

passa res. Django recalcula els números des de Postgres i els torna a escriure a la pissarra.

Per què importa: Velocitat. Postgres és fiable però relativament lent — emmagatzema dades en disc, cosa que implica lectura i escriptura físiques. Redis emmagatzema tot en RAM, que és ordres de magnitud més ràpid. Per a dades que canvien rarament però es llegeixen constantment, això és la diferència entre una app àgil i una que es nota feixuga.

Què sentiràs dir: "Està a la cache de Redis" vol dir que les dades s'estan servint des de la pissarra en lloc de l'arxivador. "La cache està obsoleta" vol dir que la pissarra no s'ha actualitzat i mostra números antics. "Esborra la cache" vol dir esborrar la pissarra i deixar que es reconstrueixi des de la base de dades.

Com es comuniquen — La vida d'un clic

Aquestes quatre peces no són un gran programa. Són quatre programes separats que es comuniquen enviant-se missatges entre ells. Això és el que passa quan cliques "Afegir tasca" i escrius "Comprar queviures":

1. **React (el teu navegador):** Cliques el botó, escrius el text, prems enter. React envia un missatge a Django: "Nova tasca: Comprar queviures, al projecte #3, de l'usuari #42."
2. **Django (el backend):** Rep el missatge. Comprova — l'usuari #42 està connectat? L'usuari #42 pertany al projecte #3? El títol no és buit i té menys de 500 caràcters? Tot correcte. Diu a Postgres que l'emmagatzemi.
3. **Postgres (la base de dades):** Crea una nova fila: Tasca #248, "Comprar queviures", Projecte #3, creada ara, estat

"Pendent". Diu a Django: "Fet, aquí tens la nova tasca amb el seu ID."

4. **Django diu a Redis:** "Incrementa el recompte de tasques del projecte #3 de 47 a 48." Redis actualitza la seva pissarra.
5. **Django diu a React:** "Aquí tens la nova tasca. S'ha desat. Tasca #248." React afegeix la targeta al teu tauler.

Tot això passa en uns 200 mil · lisegons. Menys temps del que trigues a parpellejar.

Si alguna cosa falla — posem que Postgres està caigut — Django envia un error a React, i React et mostra un missatge amable: "Alguna cosa ha anat malament, si us plau torna-ho a provar." La cuina s'ha incendiat, però el cambrer no llança fum al menjador. Dóna la mala notícia educadament.

L'API — La nota de comanda

Els missatges entre React i Django segueixen un format específic. Pensa-hi com una nota de comanda estandarditzada al restaurant — el cambrer escriu les comandes en un format que la cuina pot llegir, i la cuina envia el menjar en plats que el cambrer pot portar.

Aquest format s'anomena **API** — Application Programming Interface. És un contracte. React sap exactament com preguntar, Django sap exactament com respondre. Si React envia una petició que no coincideix amb el contracte — com demanar un plat que no és al menú — Django retorna un error.

Sentiràs la gent dir coses com "l'API retorna un 500". Aquest número és un codi d'estat — una abreviatura del que ha passat:

- **200** — Tot bé. Aquí tens el teu menjar.
- **401** — No estàs connectat. Qui ets?

- **403** — Estàs connectat, però no tens permís.
- **404** — Això no existeix. (Aquest el coneixes — "404 Not Found.")
- **500** — Alguna cosa ha explotat a la cuina. És culpa nostra, no teva.

Quan el teu amic desenvolupador diu "l'API està caiguda", vol dir que el canal de comunicació entre el frontend i el backend ha deixat de funcionar. El menjador està bé. La cuina està bé. Però la porta entre ells s'ha encallat.

Autenticació — La polsera

Com sap Django que ets *tu* qui fa la petició?

Quan inicies sessió — escrius el teu correu i contrasenya, o cliques "Inicia sessió amb Google" — Django verifica la teva identitat i dóna al teu navegador un **token**. Pensa-hi com una polsera en un festival. Durant la resta de la sessió, cada missatge que React envia a Django inclou aquesta polsera. Django hi fa un cop d'ull i diu: "Sí, aquest és l'usuari #42, té permís per ser aquí."

Si la polsera caduca (la majoria de tokens duren unes hores o dies), et retornen a la pantalla d'inici de sessió. Si algú et roba la polsera — és una bretxa de seguretat. Per això "tanca la sessió en ordinadors compartits" no és només un suggeriment.

Quan el teu desenvolupador diu "l'auth està trencada", vol dir que aquest sistema de polseres té un problema. Potser els tokens no s'estan emetent. Potser caduquen massa ràpid. Potser no s'estan comprovant correctament, cosa que és molt més preocupant — vol dir que el vigilant no fa la seva feina.

WebSockets — El walkie-talkie

La majoria de la comunicació web funciona com enviar cartes. React envia una petició, espera, i Django envia una resposta. S'anomena petició-resposta, i és com funciona la major part d'internet.

Però què passa amb les funcions en temps real? Quan un company d'equip afegeix una targeta al tauler, vols veure-ho *immediatament* — no la pròxima vegada que refresquis la pàgina. Aquí és on entren els **WebSockets**.

Un WebSocket és com un walkie-talkie. En lloc d'enviar cartes d'anada i tornada, React i Django obren una *connexió persistent* — una línia que es manté oberta. En el moment que alguna cosa canvia al servidor, Django ho emet a tothom que està escoltant. El teu tauler s'actualitza en temps real. Veus el cursor movent-se mentre el teu company arrossega una targeta.

Així és com funcionen les apps de xat. Per això pots veure "La Sara està escrivint…" en temps real. És un WebSocket — una línia oberta entre el teu navegador i el servidor, escoltant constantment.

Git — El botó de desfer per a tot

Figura 4: *Cada experiment té la seva còpia segura.*

Abans d'aprofundir més en la infraestructura, has de conèixer Git. No perquè l'utilitzaràs directament cada dia — sinó perquè és la raó principal per la qual pots deixar que els agents treballin en el teu codi sense perdre la son.

Imagina que estàs escrivint un document llarg a Google Docs. Coneixes aquella funció d'"Historial de versions"? On pots veure cada canvi que ha fet qualsevol persona, i restaurar qualsevol versió anterior? Git és això — però molt, molt més potent. És com pràcticament tot el programari del món es construeix.

Els conceptes bàsics

Repositori (repo): Una carpeta de projecte que ho recorda tot. Cada fitxer, cada canvi, cada versió — tot des del primer dia. El teu clon de Trello viu en un repo.

Commit: Una instantània. Com guardar la partida. "Aquí tens com era tot a les 3 de la tarda de dimarts." Cada commit té un missatge curt descrivint què ha canviat: "Afegida calculadora de preus" o "Arreglada la transparència del vidre." Pots tornar a qualsevol instantània, en qualsevol moment.

Branch: Un univers paral · lel. Dius: "Vull provar d'afegir una previsualització 3D, però no estic segur que funcionarà." Crees un branch — una còpia del teu projecte on pots experimentar lliurement. Si funciona, el fusiones de nou al projecte principal. Si no, elimines el branch. L'original no s'ha tocat.

Pensa-hi com dibuixar sobre paper de calc posat sobre el teu plànol. Experimenta tot el que vulguis. El plànol de sota no canvia fins que decideixis fer-ho permanent.

Merge: Agafar el paper de calc i premsar-lo sobre el plànol. L'experiment ha funcionat, així que incorpores aquests canvis al projecte principal.

Pull Request (PR): Abans de fer el merge, pots demanar a algú que revisi el teu paper de calc. "Ei, mira el que he canviat — et sembla bé?" Aquest pas de revisió és un pull request. És on els companys d'equip — o tu, revisant la feina d'un agent — examinen els canvis abans que es facin oficials.

GitHub: El lloc on els repos viuen en línia. Pensa-hi com Google Drive per a codi. El teu repo s'emmagatzema allà, amb còpia de seguretat i compartible. També és on passen els pull requests i les revisions de codi.

Per què Git importa per treballar amb agents

Aquí tens per què això és tan important. Quan dius a un agent "afegeix una previsualització 3D amb Three.js", l'agent pot canviar quinze fitxers. Potser funciona meravellosament. Potser ho

trenca tot. Sense Git, estaries atrapat — Ctrl+Z no funciona a través de quinze fitxers.

Amb Git, dius a l'agent: treballa en un branch. Ara tots els seus canvis estan continguts en aquell univers paral·lel. Ho proves. Si la previsualització 3D queda genial — merge. Si l'agent s'ha descontrolat i ha reescrit tota la lògica de preus sense motiu — elimina el branch. Cinc segons. Cap dany.

El flux és així:

```
main (la teva app estable)
  |
  |-- branch: "add-3d-preview"
  |     |-- L'agent fa canvis (commit: "Add Three.js
scene")
  |     |-- L'agent arregla un bug  (commit: "Fix camera
angle")
  |     |-- Ho revises -- queda bé!
  |     +-- Merge de tornada a main
  |
  |-- main ara té la previsualització 3D
  |
  |-- branch: "pdf-generation"
  |     |-- L'agent prova alguna cosa (commit: "Add PDF
export")
  |     |-- És un desastre -- els PDFs surten en blanc
  |     +-- Elimina el branch. Main no s'ha tocat.
  |
  +-- main segueix segur. Torna-ho a provar.
```

Cada experiment és segur. Cada error és reversible. Aquell diagrama de branches és la teva xarxa de seguretat, dibuixada. Els enginyers viuen en aquest flux cada dia. Ara entens per què.

Git vol dir que sempre pots tornar enrere. Això vol dir que pots ser audaç.

DNS — La guia telefònica d'internet

Figura 5: *La guia telefònica d'internet connecta noms amb números.*

Escrius `trello.com` al teu navegador. Però els ordinadors no entenen noms — entenen números. En algun lloc, hi ha un sistema de cerca gegant que tradueix `trello.com` a `104.192.141.1` — l'adreça real del servidor on viu Trello.

Aquest sistema és el **DNS** — el Domain Name System. Funciona exactament com els contactes del teu telèfon. Toques "Mare", el teu telèfon sap que vol dir +45 12 34 56 78. Mai penses en el número. Però sense ell, la trucada no connecta.

Quan el teu amic desenvolupador diu "el DNS encara no s'ha propagat", vol dir: hem canviat el número de telèfon, però no tothom té la guia actualitzada. Els canvis de DNS poden trigar de minuts a hores a estendre's per internet. Per això, després de llançar un lloc web nou, algunes persones el poden veure i altres no — la seva guia encara mostra l'adreça antiga.

Cada lloc web que has visitat mai ha començat amb una consulta DNS. Simplement no t'hi has fixat, perquè triga uns 20 mil · lisegons.

VPS — El pis on viu la teva app

El teu backend Django, la teva base de dades Postgres, la teva cache Redis — necessiten un ordinador on funcionar. No el teu portàtil. Un ordinador que està encès les 24 hores del dia, 7 dies a la setmana, connectat a internet, en un edifici amb energia de reserva i un aire condicionat de debò.

Un **VPS** — Virtual Private Server — és llogar una porció d'un d'aquests ordinadors. No tens tota la màquina. Tens una secció aïllada, amb el teu propi sistema operatiu, el teu propi emmagatzematge, la teva pròpia memòria. Com llogar un pis en lloc de comprar una casa — l'edifici és compartit, però el teu pis és teu.

Noms habituals que sentiràs: **DigitalOcean**, **Hetzner**, **Linode**, **AWS**, **Google Cloud**. Els tres primers són com llogar un pis senzill — aquí tens el teu servidor, aquí tens la clau, bona sort. AWS i Google Cloud són més com ciutats senceres — milers de serveis, aclaparador si només necessites on executar la teva app, però potents si necessites escalar a milions d'usuaris.

Quan algú diu "el servidor està caigut", vol dir que aquest ordinador — el VPS — ha deixat de respondre. Quan diuen "hem d'escalar", vol dir que el pis és massa petit i necessiten un de més gran, o diversos.

React és una mica diferent dels altres: es descarrega al *teu* navegador i s'executa al *teu* ordinador. Django, Postgres i Redis viuen tots al servidor. React és l'única part que viatja fins al client.

Migracions — Remodelar la base de dades

Què passa quan vols afegir un camp "data de venciment" a les tasques? No pots simplement afegir una columna nova a l'arxivador. Postgres necessita una instrucció formal: "Afegeix una nova columna anomenada `due_date` al calaix de tasques."

Aquesta instrucció s'anomena una **migració**. Django la genera automàticament quan canvies el teu model de dades, i remodela la base de dades sense perdre cap dada existent. Cada fila del calaix de Tasques rep un nou espai buit per a "data de venciment", i a partir d'ara, les noves tasques poden incloure una data de venciment.

Les migracions també poden eliminar columnes, reanomenar-les o canviar-ne el tipus. S'apliquen en ordre, com un registre de canvis — migració 001, 002, 003. Pots veure l'historial complet de com la teva base de dades va evolucionar d'una sola taula a cent.

Quan el teu desenvolupador diu "hem d'escriure una migració per a això", vol dir que l'estructura de la base de dades ha de canviar per suportar una nova funcionalitat. Sona intimidant, però és rutinari — la majoria d'apps executen dotzenes o centenars de migracions al llarg de la seva vida.

CI — La llista de comprovació automatitzada

Un desenvolupador acaba un canvi. Potser ha arreglat un bug. Potser ha afegit una nova funcionalitat. I ara què? No pot simplement enganxar-ho al servidor en producció i esperar que tot vagi bé. (Pot fer-ho. Els enginyers tenen un dit: mai facis deploy un divendres.)

CI — Continuous Integration — és una llista de comprovació automatitzada que s'executa cada vegada que algú proposa un canvi. Pensa-hi com un control de qualitat a la fàbrica abans que el producte surti de l'edifici:

1. **El codi s'executa sense errors?** De vegades un canvi trenca alguna cosa òbvia — una errada tipogràfica, un fitxer que falta, una versió incompatible.
2. **Passen tots els tests?** L'app té centenars de tests automatitzats — petits scripts que verifiquen coses com "inicia sessió com a usuari, crea una tasca, marca-la com a feta, verifica que apareix a la columna de Fets." Si algun d'aquests escenaris falla, CI ho detecta.
3. **Segueix les regles d'estil de l'equip?** Format consistent, convencions de noms, sense codi de depuració oblidat.
4. **Introdueix problemes de seguretat?** Vulnerabilitats conegudes en dependències, credencials exposades, patrons insegurs.

Si alguna cosa falla, el canvi es rebutja amb una marca vermella i un missatge explicant què s'ha trencat. El desenvolupador ho arregla i ho torna a intentar.

Si tot passa — marca verda. Ara un company revisa el canvi en un pull request, i si l'aprova, CI pot automàticament **desplegar-lo** — pujar la nova versió al VPS. La pròxima vegada que algú carregui l'app, obtindrà la versió actualitzada.

Tot el procés dura minuts, s'executa sense que cap humà hi toqui, i detecta problemes que els humans passen per alt. És la raó per la qual les apps s'actualitzen constantment sense que te n'adonis — entre bastidors, els canvis flueixen per aquest pipeline dotzenes de vegades al dia.

Eines habituals de les quals sentiràs parlar: **GitHub Actions**, **GitLab CI**, **Jenkins**. Totes fan aproximadament el mateix — executar la llista de comprovació, informar del resultat.

Mantenir el pols

No necessites escriure codi per estar al dia del que passa al món del programari. Les millors persones no tècniques amb qui he treballat tenen una cosa en comú: saben què hi ha per allà fora. Llegeixen sobre una nova eina a Hacker News tres setmanes abans que l'equip d'enginyeria la mencioni. Descobreixen un projecte de codi obert en tendència a GitHub que resol exactament el problema del qual l'equip es queixava. No són experts en aquestes eines — però saben que existeixen, i això ja és mig camí.

Tres llocs que val la pena visitar unes quantes vegades per setmana:

Hacker News (news.ycombinator.com) — Un agregador d'enllaços gestionat per Y Combinator, la incubadora de startups darrere d'Airbnb, Dropbox i Stripe. Els enginyers hi publiquen i discuteixen articles sobre tecnologia, eines i canvis en la indústria. Les seccions de comentaris són or — hi veuràs enginyers reals discutint si una nova eina és realment bona o només màrqueting. Dona un cop d'ull als cinc articles principals unes quantes vegades per setmana i estaràs per davant de la majoria de gent a qualsevol reunió.

GitHub Trending (github.com/trending) — Una llista diària i setmanal de projectes de codi obert que guanyen tracció. No entendràs tot el codi — no cal. Llegeix les descripcions dels projectes. "Una alternativa més ràpida a Elasticsearch." "Clon de Notion auto-allotjat." "Framework d'agents d'IA per a Python."

Comences a veure patrons: quins problemes resol la gent, què guanya impuls, què pot adoptar el teu equip el pròxim trimestre.

Reddit (r/programming, r/webdev, r/selfhosted, r/artificial) — Discussions de comunitat, més informals que Hacker News, sovint més pràctiques. La gent comparteix què està construint, què s'ha trencat, què ha après. Genial per a la perspectiva "com se sent realment la gent respecte a això?" Quan surt una nova eina d'IA, Reddit et dirà en 24 hores si és genuïnament útil o només màrqueting.

L'hàbit és senzill. Deu minuts amb el cafè del matí, tres vegades per setmana. Dona un cop d'ull a la portada. Llegeix un article que et cridi l'atenció. En un mes, començaràs a reconèixer noms — frameworks, biblioteques, empreses. En tres mesos, sentiràs el teu amic desenvolupador parlar d'alguna cosa i pensaràs: "Oh, vaig llegir sobre això la setmana passada."

Aquell moment — quan deixes de ser la persona que assenteix a les reunions i passes a ser la persona que fa la pregunta que canvia la direcció de la conversa — és quan aquest capítol ha fet la seva feina.

Ara ja ho saps

Això és tot. Quatre blocs bàsics. Un menjador (React), una cuina (Django), un rebost (Postgres) i una zona de preparació (Redis). A més de la infraestructura que ho manté tot unit: Git per al control de versions, DNS per trobar l'app, un VPS per executar-la i CI per assegurar-se que res es trenca quan surten canvis.

Gairebé cada aplicació web que fas servir diàriament — el teu banc, el teu servei de streaming, les teves eines de gestió de

projectes — és alguna variació d'aquest patró. Els noms específics canvien. La forma no.

Ara, quan el teu amic desenvolupador digui "l'API retorna un 500" o "hem d'afegir una cache Redis per a això" o "estic escrivint una migració", sabràs quina part del restaurant està tenint un mal dia. I quan et posis amb un agent i diguis "construeix-me una eina de pressupostos" — entendràs què està construint realment.

Aquesta comprensió és la teva targeta d'embarcament per a la resta d'aquest llibre.

Què és un agent, realment?

Figura 6: *Observar, planificar, actuar, comprovar — repetir.*

Fa anys que fas servir agents. Simplement no ho sabies.

Quan la teva app de correu mou un butlletí a la pestanya de Promocions, això és un agent — un de petit, amb una feina molt concreta. Quan el teu telèfon et suggereix "normalment surts cap a la feina a les 8:15, hi ha molt de trànsit, surt ara" — això és un agent. Quan Spotify et crea una llista de reproducció basada en el que has estat escoltant — agent.

Aquests són agents petits. D'una sola tasca. Limitats. Observen alguna cosa (el teu correu, la teva ubicació, el teu historial d'escolta), decideixen alguna cosa (això és una promoció, el trànsit és dolent, t'agradaria aquesta cançó) i actuen (mouen el correu, envien una notificació, posen la cançó a la cua).

Els agents dels quals tracta aquest llibre són la mateixa idea, però molt, molt més grans. No només classifiquen el teu correu — poden llegir tot el teu codi, planificar una sèrie de canvis, escriure el codi, executar els tests i iterar sobre els errors. No fan una cosa enginyosa. Fan *moltes* coses, en seqüència, adaptant-se a mesura que avancen.

El bucle

Cada agent, des del filtre de correu més senzill fins a l'eina d'escriptura de codi més sofisticada, executa el mateix bucle bàsic:

1. **Observar.** Recopilar informació. Llegir els fitxers. Veure el missatge d'error. Mirar l'estat actual de les coses.
2. **Planificar.** Decidir què fer. "El test falla perquè aquesta funció retorna el valor incorrecte. He de canviar la línia 47."
3. **Actuar.** Fer l'acció. Escriure el codi. Moure el fitxer. Enviar el missatge.
4. **Comprovar.** Ha funcionat? Executar el test. Llegir la sortida. Veure si l'error ha desaparegut.
5. **Repetir.** Si ha funcionat, passar a la tasca següent. Si no, tornar al pas 1 amb nova informació.

Això és tot. Observar, planificar, actuar, comprovar, repetir. El bucle és el mateix tant si l'agent està ordenant la teva bústia d'entrada com si està construint una aplicació web. El que canvia és l'*abast* — quant pot veure l'agent, quantes eines té i quanta autonomia li has donat.

Un agent simple executa aquest bucle una vegada. Un agent potent l'executa dotzenes de vegades, ajustant el seu enfocament cada cop, construint sobre el que ha après dels intents anteriors.

Les eines defineixen la capacitat

Un agent sense eines és simplement un chatbot. Pot *pensar* i *respondre*, però no pot *fer* res. Les eines que dones a un agent defineixen de què és capaç.

Un agent amb accés al teu sistema de fitxers pot llegir i escriure codi. Un agent amb un navegador web pot investigar documentació. Un agent amb un terminal pot executar comandes, llançar tests i engegar servidors. Un agent amb accés al teu repositori Git pot crear branches, fer commits i obrir pull requests.

Pensa-hi com un empleat nou. Una persona intel · ligent sense accés a cap sistema és simplement algú assegut davant un escriptori buit. Dóna-li accés a l'eina de gestió de projectes i pot fer seguiment de la feina. Dóna-li accés al repositori de codi i pot fer canvis. Dóna-li credencials de desplegament i pot posar coses en producció.

Les eines són l'accés. L'agent és la persona. Tu decideixes quin accés concedir en funció del que confies que pugui fer — que és exactament del que tracta el capítol del Gradient de Confiança.

Què no és un agent

Un agent no és una persona. No té objectius, desitjos ni sentiments. No està "intentant" ajudar-te en cap sentit significatiu. Està executant un procés de reconeixement de patrons molt sofisticat — predient la següent acció útil més probable donada tota la informació que pot veure.

Aquesta distinció importa perquè canvia com hi treballes. No motives un agent. No necessites ser amable (tot i que no fa mal). El que necessites ser és *clar*. Un agent respon a la claredat de

la mateixa manera que una persona respon a la motivació. Com millor descriguis el que vols, millor serà el resultat. Sempre.

Un agent tampoc és infal · lible. Produirà respostes incorrectes amb total confiança. Al · lucinarà biblioteques que no existeixen. Resoldrà el problema equivocat amb una precisió preciosa. Treballarà incansablement en la direcció equivocada tret que li corregeixis el rumb.

Aquesta és la teva feina. No escriure el codi — *dirigir*.

L'espectre

No tots els agents són iguals. Hi ha un espectre des del simple fins al sofisticat:

Autocompletar — La forma més simple. Comences a escriure i l'eina prediu què ve a continuació. GitHub Copilot ho fa per al codi. El teclat del teu telèfon ho fa per al text. Poca autonomia, poc risc, constantment supervisat.

Assistents de xat — Fas una pregunta, obtens una resposta. ChatGPT, Claude, Gemini. Més capaços, però encara reactius — fan el que demanes, un intercanvi a la vegada. Sense eines, sense memòria entre sessions (normalment), sense capacitat d'actuar sobre el món.

Agents equipats amb eines — La mateixa intel · ligència, però amb mans. Poden llegir fitxers, executar comandes, cercar a la web, interactuar amb APIs. Aquí és on les coses es tornen potents — i on viu la resta d'aquest llibre. Claude Code, Cursor, Windsurf — aquests són agents amb accés al teu projecte, al teu terminal, a les teves eines.

Agents autònoms — Agents que funcionen sense supervisió durant períodes llargs. Els dones un objectiu, ells descobreixen els passos, els executen, gestionen els errors i tornen amb resultats. Els més potents i els més perillosos. La majoria de les mesures de seguretat d'aquest llibre existeixen per aquesta categoria.

On et situïs en aquest espectre depèn del teu nivell de comoditat, del teu cas d'ús i de quant confies en el resultat. La majoria de gent que llegeix aquest llibre treballarà principalment amb agents equipats amb eines — prou potents per construir coses reals, prou supervisats per detectar errors.

Per què això t'importa

No necessites entendre com funciona internament el model de llenguatge. No necessites saber sobre transformers, caps d'atenció o predicció de tokens. Això és el motor. Tu necessites saber conduir.

El que *sí* necessites entendre:

- Els agents treballen en un bucle: observar, planificar, actuar, comprovar.
- Les eines defineixen el que poden fer. Més eines vol dir més capacitat, però també més risc.
- S'equivoquen amb confiança tan sovint com encerten amb confiança. La verificació no és opcional.
- La claredat és el teu superpoder. Com millor comuniquis, millor serà el resultat.

Si alguna vegada has dirigit persones, ja tens l'habilitat bàsica. Estàs a punt d'aprendre com aplicar-la a un membre de l'equip molt ràpid, molt literal i molt incansable que mai necessita cafè però que tampoc mai fa preguntes per aclarir dubtes.

Com donar bones instruccions

Figura 7: *La diferència entre una bona instrucció i una de dolenta.*

Aquest és el capítol més important d'aquest llibre.

Tota la resta — entendre la pila tecnològica, saber què són els agents, configurar mesures de seguretat — dóna suport a aquesta habilitat. Perquè un agent és tan bo com les instruccions que li dones. No de vegades. Sempre.

Si no t'emportes res més d'aquest llibre, emporta't això: *com expliques la feina és més important que l'eina que fa la feina.*

Dos capitans

Deixa'm explicar-te la història de dues persones que volien el mateix: un quadre de comandament senzill que mostrés els números de vendes setmanals del seu equip.

L'Àlex va obrir un agent i va escriure: "Fes-me un quadre de comandament de vendes."

L'agent va construir alguna cosa. Semblava un quadre de comandament. Tenia gràfics. Però les dades eren fictícies — números falsos, no connectats a res real. Els gràfics mostraven totals mensuals, no setmanals. No hi havia manera de filtrar per membre de l'equip. I feia servir una biblioteca de gràfics de JavaScript que l'Àlex no havia sentit mai, que no coincidia amb l'app de React que suposadament estaven construint.

L'Àlex va passar una hora intentant explicar què estava malament. Cada correcció introduïa un nou malentès. Després de dues hores, va llençar la tovallola i va començar de zero.

La Maia va obrir el mateix agent i va escriure una cosa diferent:

"Construeix una pàgina de quadre de comandament de vendes per a la nostra app de React. Ha de:

- Connectar-se al nostre endpoint de l'API Django existent a `/api/sales/`
- Mostrar un gràfic de barres dels totals de vendes de cadascuna de les últimes 8 setmanes
- Cada barra ha d'estar desglossada per membre de l'equip (gràfic de barres apilades)
- Fer servir la biblioteca Recharts, que ja tenim instal · lada
- Incloure un desplegable per filtrar per membre individual de l'equip, o mostrar tots

- La pàgina ha de coincidir amb l'estil existent de la nostra app — capçalera blava, targetes blanques, fons gris
- Posar-la a la ruta `/dashboard`"

L'agent va construir exactament això. Al primer intent. Trenta minuts incloent-hi una petita correcció al format de data.

Mateix agent. Mateixa tasca. Mateix objectiu. La diferència era totalment en les instruccions.

Les tres parts d'una bona instrucció

Cada bona instrucció té tres components: **què** vols, **per què** importa i **com sabràs que ha funcionat**.

Què vols
Sigues específic. No "fes un quadre de comandament" sinó "fes un gràfic de barres que mostri els totals de vendes setmanals". No "arregla el disseny" sinó "la barra lateral se superposa al contingut principal en pantalles de menys de 768 píxels — fes que la barra lateral es plegui en un menú d'hamburguesa."

Com més concreta sigui la teva descripció, menys haurà de suposar l'agent. I quan els agents suposen, suposen amb confiança. No preguntaran "vols dir setmanal o mensual?" Simplement triaran un i ho construiran.

Per què importa
El context ho canvia tot. "Afegeix un indicador de càrrega" està bé. "Afegeix un indicador de càrrega — els nostres usuaris amb connexions rurals lentes veuen una pantalla en blanc durant 3-4 segons i pensen que l'app està trencada" és millor. Ara l'agent entén el *problema*, no només la funcionalitat. Podria suggerir solu-

cions addicionals que no havies considerat, com skeleton screens o càrrega optimista.

Com sabràs que ha funcionat
Aquesta és la que la majoria de gent se salta. "Construeix una pàgina d'inici de sessió" no dóna a l'agent cap manera de verificar la seva pròpia feina. "Construeix una pàgina d'inici de sessió — hauria de poder introduir un correu electrònic i una contrasenya, clicar Iniciar sessió i ser redirigit al quadre de comandament. Si introdueixo una contrasenya incorrecta, hauria de veure un missatge d'error que digui 'Credencials no vàlides.' El botó d'inici de sessió hauria d'estar desactivat mentre la petició està en curs."

Ara l'agent té *criteris d'acceptació*. Pot comprovar la seva pròpia feina contra les teves expectatives. Aquest és el mateix concepte que fan servir els enginyers quan escriuen tests — definir com és l'èxit *abans* de construir-ho.

Les restriccions són la meitat de la feina

Dir a un agent què ha de fer és només la meitat de la feina. Dir-li què *no* ha de fer és igual d'important.

Els agents són entusiastes. Optimitzen per resoldre el problema que has descrit, i redissenyaran tota la teva interfície amb molt de gust per fer-ho. Això no és malícia — és un optimitzador fent el que fan els optimitzadors. La teva feina és establir els límits.

Bones restriccions:

- "No canviïs cap pàgina existent — només afegeix la nova pàgina de quadre de comandament."
- "Fes servir l'esquema de colors existent. No introdueixis colors nous."

- "No afegeixis noves dependències. Fes servir les biblioteques que ja tenim."
- "Manté l'estructura de fitxers igual. Posa el nou component a `src/pages/`."
- "No modifiquis l'API. El quadre de comandament ha de funcionar amb les dades que l'API ja retorna."

Pensa en les restriccions com les línies de la carretera. L'agent pot conduir lliurement dins dels carrils, però no pot creuar les línies. Sense línies, obtens solucions creatives que tècnicament funcionen però creen caos. Amb elles, obtens solucions que encaixen naturalment amb el que ja existeix.

Com més experiència adquireixis, més les teves instruccions estaran definides per les restriccions. Aprens quines llibertats porten a bons resultats i quines porten a un agent reescrivint tota la teva biblioteca de components perquè li has demanat que arregli el color d'un botó.

Els nivells d'instrucció

Hi ha un espectre des del vague fins al precís, i saber on situar-se és una habilitat que es desenvolupa amb el temps:

Nivell 0 — El desig: "Fes que l'app sigui millor." Això no dóna res a l'agent amb què treballar. O no farà res útil o ho canviarà tot.

Nivell 1 — L'objectiu: "Afegeix una manera perquè els usuaris exportin les seves dades." Millor — hi ha un objectiu clar. Però l'agent ha de decidir el format, la interfície, l'abast. Podries obtenir una exportació a CSV. Podries obtenir un endpoint d'API complet. Podries obtenir un botó de "Descarrega-ho tot" que exporta tota la base de dades.

Nivell 2 — L'especificació: "Afegeix un botó etiquetat 'Exportar a CSV' a la pàgina de configuració. Quan es cliqui, hauria de descarregar un fitxer CSV que contingui totes les tasques del projecte actual, amb columnes: Nom de la Tasca, Estat, Assignat, Data de Venciment, Data de Creació. El fitxer hauria de dir-se `nom-projecte-export-2026-03-18.csv`." Aquí és on vols estar per a la majoria de tasques. Prou específic perquè l'agent no pugui malinterpretar-ho, prou flexible perquè pugui gestionar els detalls d'implementació.

Nivell 3 — El plànol: Especificació completa d'implementació amb rutes de fitxers exactes, noms de funcions i patrons de codi a seguir. Normalment innecessari tret que treballis en un codi gran i complex amb convencions estrictes. El teu desenvolupador pot escriure prompts a aquest nivell. Tu probablement no ho necessitaràs.

Per a la majoria de gent que llegeix aquest llibre, el Nivell 2 és el punt ideal. Específic sobre el *què* i el *resultat*, flexible sobre el *com*.

Iterar és normal

Ningú escriu una instrucció perfecta al primer intent. Ni principiants. Ni experts. La diferència clau és que les persones amb experiència iteren *més ràpid* perquè han après quins tipus de vaguetat causen quins tipus de problemes.

Un flux de treball saludable és així:

1. Dóna a l'agent una instrucció clara.
2. Revisa el que ha produït.
3. Identifica què està malament o què falta.
4. Dóna una instrucció de seguiment *enfocada*.
5. Repeteix fins que estigui bé.

El seguiment és on la majoria de gent té dificultats. Veuen alguna cosa malament i diuen "això no està bé, arregla-ho." Això és el capità Àlex de nou. En canvi:

- "El gràfic de barres mostra totals mensuals. Canvia-ho a setmanal — agrupa per número de setmana ISO."
- "A l'exportació li falta la columna d'Assignat. Afegeix-la entre Estat i Data de Venciment."
- "L'indicador de càrrega és massa petit. Fes-lo de 48 píxels i centra'l verticalment a la targeta."

Cada seguiment és una instrucció en miniatura amb la mateixa estructura: què està malament, què vols en canvi, com verificar-ho.

Practica

Això és una habilitat, cosa que vol dir que millora amb la pràctica. Comença amb coses petites. Demana a un agent que redacti un correu electrònic i mira com d'específic has de ser sobre el to i el contingut. Demana-li que reorganitzi un full de càlcul i fixa't en com de precís has de descriure el disseny desitjat. Demana-li que planifiqui un viatge i observa on fa suposicions.

Cada vegada que l'agent produeix alguna cosa que no és el que volies, pregunta't: "Què podria haver dit diferent?" La resposta sempre és la mateixa — ser més específic sobre el que volies, més explícit sobre el que no volies, i més concret sobre com és l'èxit.

Les persones que treuen més profit dels agents no són les que tenen més coneixements tècnics. Són les que han après a comunicar-se amb precisió. I aquesta és una habilitat que es transfereix a cada conversa que tindràs mai — amb agents, amb col · legues i amb qualsevol persona que necessiti entendre què vols realment.

Què pot veure l'agent

Figura 8: *Un agent només pot treballar amb el que té al banc.*

La cosa més important de treballar amb agents no és com els dones instruccions. És què els poses al davant abans de fer la pregunta.

Un agent és tan bo com el que pot veure. Dóna-li una descripció vaga i un full en blanc, i al · lucinarà amb confiança. Dóna-li els fitxers correctes, les restriccions correctes, la visió correcta del problema — i farà coses que semblen màgia. La diferència no és el model. Ets tu.

El banc de treball

Pensa en la visió de l'agent com un banc de treball físic. Té un espai limitat. No pots abocar-hi tota la teva vida i esperar bons resultats. En canvi, hi disposes les peces que importen per a *aquesta* tasca: els documents rellevants, el missatge d'error específic, la captura de pantalla, l'exemple del que vols.

Bona gestió del banc de treball:

- Treballant en una eina de pressupostos? Posa la versió actual davant de l'agent, a més d'un exemple de com hauria de ser un pressupost acabat.
- Intentant arreglar un bug? No descriguis el bug — enganxa el missatge d'error real. Mostra la captura de pantalla. Inclou els passos que l'han provocat.
- Dissenyant una nova funcionalitat? Proporciona un esbós, l'exemple d'un competidor o una descripció detallada del flux d'usuari.

Mala gestió del banc de treball:

- Enganxar tot el teu projecte i dir "alguna cosa està trencada."
- Descriure un problema de memòria en lloc de compartir l'error real.
- Donar a l'agent informació sobre deu tasques diferents i demanar-li que endevini quina vols dir.

El principi del banc de treball s'aplica a tot — no només al codi. Si demanes a un agent que redacti un informe, dóna-li les dades en brut, l'audiència, el format que vols i un exemple d'un bon informe. Si li demanes que planifiqui un esdeveniment, dóna-li les restriccions (pressupost, data, capacitat del local) en lloc de simplement "planifica un esdeveniment d'equip."

Materials en brut vs. descripcions

Hi ha una diferència crucial entre *descriure* alguna cosa i *mostrar-la*.

Descripció: "Els números de vendes estan en un full de càlcul. Els totals semblen incorrectes per al T2."

Material en brut: Enganxa les dades reals del full de càlcul. O millor — dóna a l'agent accés al fitxer. Ara pot veure les fórmules, trobar l'error i arreglar-lo. Tu has descrit el símptoma. El material en brut mostra la causa.

Aquesta és la millora més gran que la majoria de gent pot fer: deixar de descriure, començar a mostrar. Enganxa el missatge d'error. Adjunta la captura de pantalla. Puja el document. Dóna a l'agent la mateixa informació que donaries a un col·lega assegut al teu costat.

Cada vegada que parafraseges, perds senyal. Comprimeixes el problema real a través del tub estret de la teva interpretació, i l'agent l'ha de descomprimir — malament — a l'altre costat. És com descriure una pintura per telèfon i demanar a algú que la reprodueixi.

Nivells de qualitat del context

Hi ha una manera útil de pensar sobre com de bo és el teu context:

Nivell 0 — Descripció vaga: "L'informe sembla incorrecte." L'agent endevina quin informe, endevina què està malament i endevina com arreglar-ho. Amb tres suposicions de profunditat, les probabilitats d'encertar-ho són baixes.

Nivell 1 — Descripció específica: "Els ingressos del T2 a l'informe trimestral mostren 2,3 M$ però haurien de ser 2,1 M$.

Crec que la fórmula inclou les transaccions reemborsades." Molt millor. L'agent coneix el símptoma i la teva teoria sobre la causa.

Nivell 2 — Dades en brut: Enganxes la secció rellevant del full de càlcul, la fórmula i la llista de transaccions. Ara l'agent pot verificar la teva teoria o descobrir la causa real. Potser la fórmula està bé però dues transaccions estaven duplicades. No ho hauries detectat descrivint el problema.

Nivell 3 — Accés: L'agent pot obrir el full de càlcul ell mateix, executar les fórmules, comprovar la font de dades i investigar independentment. Tu proporciones la *intenció* ("esbrina per què els ingressos del T2 estan inflats") i l'agent fa la feina de detectiu.

La majoria de gent viu al Nivell 0 o 1. L'objectiu és arribar al Nivell 2 com a predeterminat, i al Nivell 3 quan les eines ho permetin.

Menys és més (de vegades)

No es tracta de donar a l'agent *tot*. Es tracta de donar-li *les coses adequades*.

Massa poc context i l'agent endevina. Massa context i l'agent queda enterrat. Imagina donar a algú una pila de mil pàgines i dir "la resposta és en algun lloc d'aquí dins." Bona sort.

Per a cada tasca, pregunta't:

- Què necessita veure l'agent per entendre aquesta tasca?
- Què el confondrà si ho incloc?
- La informació que estic proporcionant és actual, o li estic donant dades obsoletes?

Un banc de treball enfocat supera un de desordenat. Tres fitxers rellevants són millors que trenta d'irrellevants. El missatge d'error específic és millor que tot el fitxer de registre.

L'efecte compost

Cada vegada que dones a un agent bon context, el resultat és millor. Millor resultat vol dir menys temps corregint. Menys correccions vol dir més temps per a la tasca següent. Al llarg d'una setmana, la diferència entre algú que aboca descripcions vagues i algú que proporciona context net i enfocat és enorme — no perquè sigui més intel·ligent, sinó perquè ha après a preparar l'agent per a l'èxit.

Aquesta habilitat es compon. I és exactament la mateixa habilitat que et fa millor delegant a humans, escrivint correus més clars i fent millors informes de bugs. L'agent simplement et dóna retroalimentació més ràpida sobre si la teva comunicació era prou clara.

El Gradient de Confianca

Figura 9: *Comença estret. Afluixa amb evidència.*

Deixaries que un becari nou envia correus als teus clients el primer dia? Probablement no. El deixaries redactar correus perque tu els revisessis? Es clar. El deixaries enviar respostes rutinaries despres d'un mes de feina, havent vist com treballa? Segurament si.

Aquesta escalada – de "ja ho faig jo" a "fes-ho, pero jo ho reviso" a "endavant, confio en tu" – es exactament com hauries de pensar en treballar amb agents. S'anomena el gradient de confianca, i fer-ho be es la diferencia entre un agent que t'ajuda i un que crea un embolic que et passa la tarda arreglant.

La Taula de Mescles

Pensa en el teu nivell de confianca com una taula de mescles – d'aquelles amb controls lliscants. Cada tipus de tasca te el seu propi control:

- **Llegir fitxers i dades** – Risc baix. L'agent nomes mira. Puja'l.
- **Escriure i editar documents** – Risc mitja. L'agent podria canviar alguna cosa que t'importa. Deixa'l a "revisar abans de desar."
- **Enviar missatges o correus** – Risc mes alt. Un cop enviat, ja esta enviat. Deixa'l a "esborrany per a la meva aprovacio."
- **Fer compres o decisions financeres** – Risc alt. Deixa'l a "suggereix, no actueis."
- **Eliminar o sobreescriure dades** – Risc alt. Deixa'l a "pregunta'm primer."

No poses tots els controls igual, i no els ajustes un cop i te n'oblides. Es mouen amb el temps a mesura que guanyes confianca en arees concretes.

El Trinquet

El primer dia amb un agent, tot es revisa. Cada sortida es comprova. Encara no saps on es brillant i on es fragil, aixi que ho vigiles tot. Aixo es normal. Aixo es intel.ligent.

Despres d'uns dies, apareixen patrons. L'agent es excel.lent redactant informes. Es solid en analisi de dades. De vegades fa tries qüestionables sobre el to en correus de cara al client. Ara els teus controls ho reflecteixen: informes i analisis funcionen amb revisio minima, els correus de clients reben una lectura acurada abans d'enviar-los.

Despres d'unes setmanes, has vist desenes de tasques completades amb exit. Confies mes en l'agent en les arees on ha demostrat la seva valia. Les baranes de seguretat segueixen alla, pero son invisibles per al 90% de la feina rutinaria. Nomes s'activen en situacions inusuals.

Aixo es el trinquet: un ajust lent, basat en evidencies, d'apretar i afluixar. Les persones que mai afluixen les baranes acaben abandonant els agents perque "es massa feina revisar-ho tot." Les persones que afluixen massa rapid reben el correu que no s'hauria d'haver enviat, o les dades que no s'haurien d'haver eliminat.

Baranes de Seguretat a la Practica

Per a projectes de software – del tipus que podries construir amb un agent – les baranes de seguretat tenen una forma molt concreta, i ja la coneixes: les branques de Git.

Quan un agent treballa en una branca, tots els seus canvis estan aillats. Pots revisar-los en un pull request abans que toquin el projecte principal. Aixo es una barana de seguretat. L'agent te llibertat per experimentar, pero els resultats passen per la teva aprovacio abans de convertir-se en reals.

Per a tasques que no son codi, les baranes son diferents:

- **Esborranys, no enviaments.** L'agent escriu el correu. Tu l'envies.
- **Suggeriments, no decisions.** L'agent recomana l'assignacio del pressupost. Tu l'aproves.
- **Resums amb fonts.** L'agent resumeix l'informe *i et mostra d'on ve cada afirmacio*. Tu ho verifiques.
- **Desplegaments graduals.** Prova la sortida de l'agent en una tasca petita i de baix risc abans de confiar-li la gran.

El principi es sempre el mateix: mante l'huma en el circuit per a qualsevol cosa que no es pugui desfer facilment.

Els Dos Errors

Hi ha exactament dues maneres d'equivocar-se amb el gradient de confianca:

Massa estret: Revises cada sortida, tornes a comprovar cada fet, reescrius cada frase. L'agent es converteix en una manera lenta i cara de produir un esborrany que anaves a reescriure de totes maneres. T'exhaureixes supervisant i conclous que "els agents no valen la pena." Si que valen – nomes que mai has deixat moure el trinquet.

Massa fluix: Deixes que l'agent funcioni sense supervisio perque els primers resultats eren bons. L'agent envia un correu amb un error factual. Publica un informe amb una estadistica inventada. Elimina un fitxer que no tenia copia de seguretat. Conclous que "no es pot confiar en els agents." Si que es pot – nomes t'has saltat la part on verifiques abans de donar autonomia.

El punt ideal no es cap dels dos. Es un canvi gradual, basat en evidencies – revisant-ho tot al principi, afluixant a mesura que la confianca creix, pero mantenint limits ferms en les coses que realment importen.

El Teu Paper

Potser aquesta es la idea mes encoratjadora del llibre: el gradient de confianca significa que *tu* sempre tens el control. No l'agent. No la tecnologia. Tu.

Tu decideixes que pot veure l'agent. Tu decideixes que pot fer. Tu decideixes quan revisar i quan confiar. Pots apretar les baranes de seguretat en qualsevol moment. Pots eliminar una branca. Pots rebutjar un esborrany. Pots dir "de fet, aquesta me l'encarrego jo."

Un agent es una eina amb un dial de confianca. Tu tens el dial. Aixo no es una limitacio – es el disseny.

Ampliant l'Abast de la Tripulacio

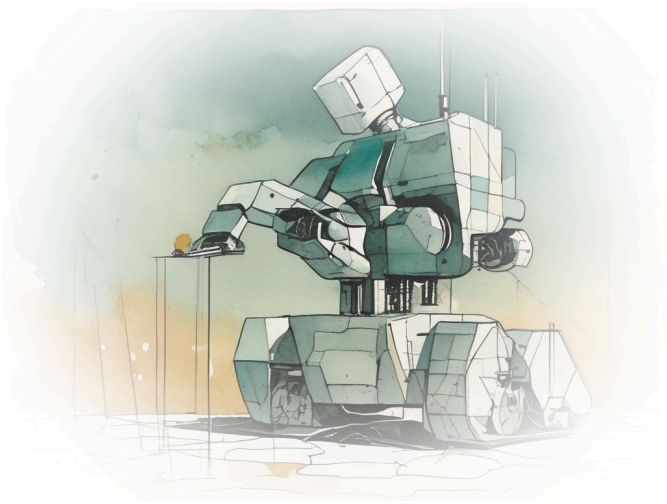

Figura 10: *Connectant l'agent amb el món més enllà de la seva finestra.*

Un agent que nomes pot llegir i escriure text es util. Un agent que es pot connectar a les teves eines – el teu calendari, els teus fulls de calcul, la teva aplicacio de gestio de projectes, el teu correu – es transformador.

Aquest capitol tracta de donar *mans* al teu agent. No la fontaneria tecnica (el teu desenvolupador se n'encarrega) sino el concepte del que es possible quan un agent pot anar mes enlla de la finestra de conversa i interactuar amb els sistemes que ja fas servir.

L'Escriptori Buit vs. L'Estacio de Treball Connectada

Recordes la metafora del nou empleat? Una persona intel.ligent sense acces a cap sistema es nomes algu assegut davant d'un escriptori buit. Dona-li acces al teu calendari i podra programar reunions. Dona-li acces al teu CRM i podra actualitzar registres de clients. Dona-li acces al teu emmagatzematge de documents i podra investigar respostes a les teves preguntes.

Els agents funcionen de la mateixa manera. Cada connexio que afegeixes – cada *integracio* – amplia el que l'agent pot fer:

- **Connectat al teu sistema de fitxers:** L'agent pot llegir els teus documents, analitzar fulls de calcul, organitzar carpetes.
- **Connectat al web:** L'agent pot investigar competidors, consultar documentacio, buscar informacio actual.
- **Connectat al teu correu:** L'agent pot redactar respostes, resumir fils, marcar missatges urgents.
- **Connectat a la teva eina de gestio de projectes:** L'agent pot crear tasques, actualitzar estats, generar informes.
- **Connectat a la teva base de dades:** L'agent pot consultar dades, generar informes i trobar patrons que et passarien per alt en un full de calcul.

Cadascuna d'aquestes es una eina a les mans de l'agent. Mes eines significa mes capacitat – pero tambe mes responsabilitat, per aixo existeix el gradient de confianca.

Fontaneria Sense Codi

No cal ser desenvolupador per connectar coses entre si. Hi ha un ecosistema creixent d'eines per connectar sistemes sense escriure codi:

Zapier, Make (anteriorment Integromat), n8n – Aquestes son plataformes d'automatitzacio. Et permeten crear fluxos de treball "si passa aixo, fes allo": "Quan aparegui una nova fila al meu full de calcul, crea una tasca a la meva eina de gestio de projectes." "Quan rebi un correu d'un client concret, resumeix-lo i publica el resum a Slack."

Aquests no son agents – son pipelines. Pero poden treballar *amb* agents. Un agent redacta un pressupost, i un flux de treball de Zapier l'envia al client i el registra al CRM. L'agent fa el pensament. El pipeline fa la fontaneria.

Apple Shortcuts, Power Automate – Versions mes lleugeres per a fluxos de treball personals. "Cada mati, fes que un agent resumeixi els meus correus no llegits i posi el resum a la meva aplicacio de Notes." Son petits, pero s'acumulen.

MCP (Model Context Protocol) – Aixo es un estandard mes nou que permet als agents connectar-se directament a eines externes d'una manera estandaritzada. Es tecnic per dins, pero el resultat es simple: en lloc de tu copiar dades *cap a* l'agent, l'agent pot anar a *buscar* dades de les teves eines. El teu desenvolupador ho configurara. Tu gaudieras dels resultats.

L'Efecte Multiplicador

Cada integracio d'eines no nomes suma – multiplica. Un agent que pot llegir les teves dades de vendes *i* accedir al teu correu *i*

consultar el teu calendari pot fer coses que cap d'aquestes connexions podria fer sola:

"Consulta les meves dades de vendes del Q2, troba els tres comptes mes grans que no s'han contactat en mes de 30 dies, redacta correus de seguiment per a cadascun, i suggereix horaris de reunio basats en la disponibilitat del meu calendari la setmana vinent."

Aixo es un sol prompt. Quatre eines. Una tasca que t'hauria pres noranta minuts, feta en cinc.

Aixo es l'efecte compost de la integracio d'eines. Cada nova connexio obre combinacions que abans no eren possibles. Les persones que mes treuen dels agents rarament son les que tenen el model mes potent – son les que tenen l'espai de treball mes connectat.

La Questio de la Privacitat

Cada eina que connectes a un agent es una porta que obres. Abans de connectar alguna cosa, pregunta't:

- **Quines dades podra veure l'agent?** Connectar el teu correu significa que l'agent pot llegir *tot* el teu correu, no nomes el fil en que estaves treballant.
- **Que pot fer l'agent amb elles?** L'acces de nomes lectura es molt diferent de l'acces de lectura i escriptura. L'agent pot nomes veure el teu calendari, o pot crear i eliminar esdeveniments?
- **On van les dades?** Quan l'agent llegeix el teu full de calcul, aquestes dades s'envien als servidors del proveidord'IA. Aixo es acceptable per a aquestes dades? Ho seria per a les dades dels teus clients?

- **Que passaria si alguna cosa anes malament?** Un agent amb acces d'eliminacio al teu sistema de fitxers pot eliminar fitxers accidentalment. Un agent amb acces d'enviament al teu correu pot enviar un correu accidentalment. Pensa en el pitjor cas.

Aixo no es una rao per evitar la integracio d'eines. Es una rao per ser intencional. Connecta el que necessites. Dona l'acces minim necessari. I aplica el gradient de confianca: comenca amb nomes lectura, passa a lectura i escriptura despres d'haver guanyat confianca.

Llegir la Sortida com un Professional

Figura 11: *Confia, però verifica.*

Un agent mai et dira que s'equivoca. No sap que s'equivoca. Produeix sortides amb la mateixa confianca tant si es perfectament precis com si esta completament inventat. La teva feina – l'habilitat que separa un usuari productiu d'agents d'un de perillos – es aprendre a llegir la sortida amb escepticisme saludable.

Aixo no va de desconfianca. Va de confianca *calibrada*. Un bon editor confia en els seus escriptors pero tot i aixi comprova els fets. Un bon gestor confia en el seu equip pero tot i aixi revisa els lliurables. Tu ets l'editor. Tu ets el gestor. L'agent es un membre

de l'equip molt rapid, molt segur de si mateix, que de vegades s'inventa coses.

Al.lucinacions

El terme tecnic es "al.lucinacio" – quan un agent produeix informacio que sona plausible pero es factualment incorrecta. Es el mode de fallada mes important d'entendre perque es el mes dificil de detectar.

Una al.lucinacio no es un error aleatori. Es un error *plausible*. L'agent no diu "els ingressos eren de color lila." Diu "els ingressos del Q2 van ser de 2,3 milions d'euros" quan la xifra real es de 2,1 milions. No cita una font que obviament no existeix – cita una que *sona* com si hagues d'existir. No inventa una afirmacio estranya – exposa alguna cosa que encaixa perfectament en la narrativa, nomes que resulta ser falsa.

Per aixo les al.lucinacions son perilloses: passen la prova de l'olfacte. Llegeixes la sortida, sona raonable, i passes endavant. L'error es propaga al teu informe, la teva presentacio, la teva decisio.

La Llista de Verificacio

Per a qualsevol sortida d'agent que importi, repassa aixo:

Numeros: D'on venen? Pots rastrejar-los fins a les dades originals? Si l'agent diu "les vendes van creixer un 15%," comprova les dades reals. No confiis en percentatges, totals o estadistiques sense verificacio.

Noms i dates: Els agents sovint es confonen lleugerament amb els detalls. La persona correcta pero la data equivocada.

L'empresa correcta pero el producte equivocat. L'estadistica correcta pero de l'any equivocat. Comprova puntualment els detalls concrets.

Afirmacions i fets: Si l'agent exposa alguna cosa com a fet, pregunta't – es alguna cosa que jo ja se que es certa, o ho estic aprenent de l'agent? Si ho estas aprenent de l'agent, verifica-ho. Una cerca rapida al web, una comprovacio amb els teus propis registres, una consulta rapida amb un company.

Fonts i citacions: Si l'agent cita una font, comprova que la font existeix i que realment diu el que l'agent afirma que diu. Els agents son famosos per citar publicacions que sonen reals amb contingut fabricat, o per citar publicacions reals pero tergiversant el que diuen.

Completesa: La sortida omet alguna cosa obvia? Els agents tendeixen a produir feina d'aspecte plausible que cobreix el 80% del que vas demanar. El 20% que falta sovint es la part que hauria revelat un problema.

La Trampa de la Confianca

Els agents expressen tot amb el mateix nivell de confianca. "La reunio es a les 3" i "la reunio es a les 3" es veuen identic tant si l'agent ho llegeix del teu calendari com si ho endevina basant-se en el teu horari habitual.

No hi ha cursiva per a la incertesa. No hi ha "crec" o "no estic segur." La sortida arriba com si fos un fet, cada vegada.

Aixo significa que no pots confiar en el *to* de l'agent per avaluar la precisio. Has de desenvolupar el teu propi sentit sobre quins tipus de sortides son probablement precises i quines necessiten comprovacio:

Confianca alta (normalment fiable):

- Formatar i reestructurar dades existents que has proporcionat
- Seguir instruccions explicites ("ordena aquesta llista alfabeticament")
- Fer calculs amb dades que li has donat
- Resumir un document que has proporcionat

Confianca mes baixa (verifica sempre):

- Afirmacions factuals sobre el mon (dates, estadistiques, esdeveniments actuals)
- Qualsevol cosa que impliqui esdeveniments recents (l'entrenament del model te una data de tall)
- Numeros concrets que no ha calculat a partir de dades que has proporcionat
- Consells legals, medics o financers
- Afirmacions sobre el que hi ha en un document que realment no ha llegit

Crear l'Habit

La verificacio no hauria de sentir-se com una carrega. Hauria de sentir-se com una correccio de proves – un repas natural i rapid de la sortida abans de fer-la servir.

Per a tasques de baix risc (redactar un missatge intern, organitzar notes), un cop d'ull rapid n'hi ha prou. Ha capturat els punts clau? El to sembla correcte? Prou bo.

Per a tasques de risc mitja (un informe per a un client, una analisi de dades), comprova els numeros i les afirmacions clau. Rastreja almenys un o dos fets fins a la seva font. Llegeix-ho tot com si ho hagues escrit algu altre i tu fossis el revisor.

Per a tasques d'alt risc (una presentacio per a la junta, una projeccio financera, un document legal), verifica-ho tot. Tracta la sortida de l'agent com un primer esborrany que necessita comprovacio de fets, no com un producte acabat. Si no ho publicaries sense comprovar-ho quan ho ha escrit un huma, no ho publiquis sense comprovar-ho quan ho ha escrit un agent.

L'objectiu no es no confiar mai en l'agent. L'objectiu es confiar-hi *adequadament* – amb entusiasme per allo que fa be, amb precaucio per allo que fa menys be, i mai a cegues.

Construir Alguna Cosa Real

Figura 12: *De la idea al prototip funcional en un cap de setmana.*

Aquest es el capitol on tot encaixa. Recorrerem la construccio d'una aplicacio real – des de la idea fins al producte funcional – fent servir un agent. Sense codi. Sense experiencia en programacio. Nomes pensament clar, bones instruccions, i tot el que has apres fins ara.

L'aplicacio es real. El sector es real. El proces es exactament el que sembla quan algu que coneix el seu domini s'asseu amb un agent i construeix alguna cosa.

La Idea

El teu oncle instal.la finestres per guanyar-se la vida. No les de Microsoft – les de vidre. Bon negoci, feina constant. Fa quinze anys que s'hi dedica i coneix cada model de finestra, cada tipus de marc, cada truc per aconseguir un segellat perfecte en una casa vella i torta.

Pero el seu proces de pressupostos esta ancorat al 2005. Mesura la paret, tria una finestra d'un cataleg de paper, calcula el preu amb una calculadora, ho escriu tot en un document de Word, el desa com a PDF i l'envia per correu al client. La meitat del temps, el client no pot imaginar com quedara realment la finestra a la seva paret. L'altra meitat, perden el PDF i truquen demanant-ne un altre.

Li construiras alguna cosa millor. Una aplicacio on introdueix les mesures, tria un model de finestra, veu una previsualitzacio 3D de com quedara, i genera un pressupost professional – tot en un sol lloc, accessible des de la seva tauleta al lloc de treball.

Ho construiras aquest cap de setmana.

Abans de Tocar l'Agent

L'error mes gran que comet la gent es obrir l'agent i escriure immediatament "construeix-me una app de pressupostos de finestres." Aixo es el Capita Alex. Nosaltres serem la Capitana Maya.

Abans d'escriure un sol prompt, has de pensar com una persona de producte. Pregunta't:

Qui ho fara servir? Instal.ladors de finestres, amb un portatil o tauleta a casa del client. Ha de ser simple, rapid i funcionar en una pantalla de tauleta.

Quina es l'accio principal? Introduir mesures, veure la previsualitzacio, enviar el pressupost. Aixo es el flux. Tot el demes es secundari.

Quines dades hem d'emmagatzemar? Clients (nom, adreca, telefon). Models de finestra (nom, dimensions, preu per metre quadrat). Pressupostos (client, model de finestra, mesures, preu total, data).

Quin es el moment "uau"? La previsualitzacio 3D. El client esta al seu menjador i veu, en una tauleta, exactament com quedara la nova finestra a la seva paret. Aixo es el que ho fa millor que un document de Word.

Acabes de dissenyar una aplicacio. Sense codi. Sense titol tecnic. Nomes pensament clar sobre qui necessita que.

Dividir-ho en Tasques

Ara dividim el projecte en peces. Cada peca es una feina autocontinguda que donaras a l'agent. El principi clau: cada feina ha de tenir sentit per si sola, produir un resultat verificable, i construir sobre l'anterior.

Tasca 1: Configurar el Projecte
Aquests son els fonaments. Estas dient a l'agent que construeixi l'edifici buit abans de comencar a mobiliar les habitacions.

"Crea un nou projecte amb un backend de Django i un frontend de React. Configura una base de dades Postgres amb tres models: Customer (nom, email, telefon, adreca), WindowModel (nom, rang d'amplada, rang d'alcada, tipus de vidre, preu per metre quadrat), i Quote (vinculat a un client i un model de finestra, amplada de paret, alcada de paret, amplada de finestra, alcada

de finestra, posicio x de finestra, posicio y de finestra, preu total, data de creacio). Crea un panell d'administracio de Django perque pugui afegir models de finestra manualment. Assegura't que l'aplicacio React pugui comunicar-se amb el backend de Django a traves d'una API."

Aixo es un sol prompt. Configura tota la pila del Capitol 3 – el menjador, la cuina i el rebost. L'agent creara desenes de fitxers, configurara la base de dades, muntara la API i ho connectara tot.

Quan acabi, hauries de poder obrir el panell d'administracio de Django al navegador, afegir un model de finestra i veure'l retornat des de la API. Aquesta es la teva verificacio. Si aixo funciona, els fonaments son solids.

Tasca 2: El Formulari de Pressupost
Ara construim la interficie on l'instal.lador introdueix la informacio.

"Construeix una pagina a l'aplicacio React amb un formulari per crear un nou pressupost. El formulari ha de tenir: un desplegable per seleccionar un client existent o un boto per afegir-ne un de nou, un desplegable per seleccionar un model de finestra, camps numerics per a amplada de paret, alcada de paret, amplada de finestra, alcada de finestra, i posicio de finestra (horitzontal i vertical, mesurada des de la cantonada inferior esquerra de la paret). Totes les mesures han de ser en centimetres. Quan l'usuari ompli les mesures i seleccioni un model, mostra el preu calculat a sota del formulari: area de la finestra en metres quadrats multiplicada pel preu per metre quadrat del model, mes una tarifa fixa d'instal.lacio de 1.500 DKK. Afegeix un boto Desar Pressupost que enviia les dades a Django."

Fixa't en el nivell de detall. Has especificat els camps, la unitat, la formula de preus i la interaccio. L'agent no ha d'endevinar res.

Tasca 3: La Previsualitzacio 3D

Aquesta es la peca estrella – i la part que sona mes espantosa per a algu que no programa. No ho es. No estas escrivint codi de Three.js. Estas descrivint una escena.

"Utilitzant la llibreria Three.js, afegeix un panell de previsualitzacio 3D al costat del formulari de pressupost. La previsualitzacio ha de mostrar:

- Una paret, representada com un rectangle beix pla, utilitzant l'amplada i alcada de paret del formulari
- Un retall rectangular a la paret on va la finestra, posicionat segons els valors de posicio i mida de finestra del formulari
- Un vidre al retall – lleugerament transparent, amb un tint blau clar
- Un marc de finestra simple al voltant del vidre, gris fosc, de 5 centimetres de gruix
- La camera ha de comencar amb un lleuger angle perque es pugui veure que la paret te una mica de profunditat (fes la paret de 20cm de gruix)
- L'usuari ha de poder girar la vista arrossegant amb el ratoli (utilitza OrbitControls)
- Afegeix llum ambiental suau i una llum direccional des de la part superior dreta perque el vidre tingui un reflex subtil
- L'escena s'ha d'actualitzar en temps real mentre l'usuari canvia les mesures al formulari

Posa el formulari a l'esquerra de la pantalla i la previsualitzacio 3D a la dreta. En pantalles de tauleta, apila'ls verticalment amb la previsualitzacio a dalt."

Acabes de descriure una escena 3D. Com dir-li a un escenograf que vols a l'escenari. L'agent coneix Three.js. La teva feina es

saber com ha de semblar una instal.lacio de finestra – i *aixo*, el teu oncle t'ho pot dir amb els ulls tancats.

Tasca 4: El PDF del Pressupost
"Quan l'usuari faci clic a Desar Pressupost, genera un PDF amb: una capcalera amb el nom de l'empresa 'Hansen Vinduer' i el logotip (proporcionare el fitxer del logotip mes tard), el nom i l'adreca del client, una taula amb les especificacions de la finestra (model, dimensions, tipus de vidre), la previsualitzacio 3D renderitzada com a imatge estatica, el desglossament del preu (area de finestra, preu per metre quadrat, tarifa d'instal.lacio, total), i un peu de pagina amb la informacio de contacte de l'empresa i el numero de CVR. Utilitza la llibreria WeasyPrint per a la generacio de PDF al costat de Django. Despres de desar, mostra un boto 'Descarregar PDF' i un boto 'Enviar per Correu al Client'."

Tasca 5: Historial de Pressupostos
"Afegeix una pagina que llisti tots els pressupostos anteriors. Mostra'ls en una taula amb columnes: data, nom del client, model de finestra, preu total i estat (esborrany o enviat). L'usuari ha de poder fer clic a qualsevol fila per obrir el pressupost complet. Afegeix un quadre de cerca que filtri per nom de client. Ordena per data, del mes recent al mes antic."

Tasca 6: Poliment
"Dona estil a tota l'aplicacio perque es vegi neta i professional. Utilitza un esquema de colors de blau mari (hex 1a365d) per a la capcalera i els botons, blanc per als fons de les targetes, gris clar (hex f7f7f7) per al fons de la pagina. Fes que tots els botons i camps de formulari siguin prou grans per tocar facilment en una tauleta. Afegeix el nom de l'empresa 'Hansen Vinduer' a la capcalera. Assegura't que tot funcioni en un iPad en orientacio horitzontal."

El Moment Three.js

Fixem-nos en la Tasca 3, perque il.lustra la llico mes important d'aquest capitol: pots dirigir un agent perque utilitzi tecnologia de la qual mai has sentit a parlar.

Three.js es una llibreria que dibuixa grafics 3D en un navegador web. Mai l'has utilitzat. Probablement mai n'has sentit a parlar. I no necessites aprendre-la. Necessites aprendre a *descriure el que vols en 3D*.

L'agent coneix la llibreria. Tu coneixes el domini. Aquesta combinacio es mes potent que qualsevol de les dues sola.

Quan arribi la primera versio, no sera perfecta. Potser la paret esta al reves – estas mirant la cara interior en lloc de l'exterior. Potser el vidre es massa opac. Potser la camera comenca dins la paret, mirant cap a fora.

Aixo es normal. Iteres:

- "La paret mostra la cara del darrere. Gira-la perque vegem la cara exterior."
- "El vidre es massa fosc. Fes-lo un 80% transparent amb nomes un toc de blau."
- "La camera comenca massa a prop. Mou-la enrere perque pugui veure tota la paret."
- "Quan canvio l'amplada de la finestra, el marc no es redimensiona. Fes que el marc s'actualitzi en temps real mentre escric."

Cada correccio li pren a l'agent uns trenta segons. Estas dirigint, no depurant. Ets el client assegut amb l'arquitecte, dient "mou aquella finestra una mica a l'esquerra." L'arquitecte fa el dibuix.

Que et Trobaras pel Cami

Siguem honestos sobre els obstacles, perque formen part del proces:

L'agent fara suposicions. A la Tasca 1, podria configurar la base de dades de manera diferent del que esperaves. Potser posa el preu al pressupost en lloc de calcular-lo. Revisa la sortida. Si alguna cosa no quadra, digues-ho.

Les coses es trencaran entre tasques. La Tasca 2 podria no connectar-se correctament amb el que la Tasca 1 va crear. El endpoint de la API podria tenir un nom diferent del que el frontend espera. Aixo es normal en el desenvolupament de software – s'anomena *integracio*. Digues-li a l'agent: "El formulari intenta enviar dades a `/api/quotes/` pero el endpoint de la API es a `/api/quote/create/`. Corregeix el frontend perque faci servir el endpoint correcte."

La previsualitzacio 3D es veura estranya al principi. Els grafics 3D son delicats. Il.luminacio, angles de camera, propietats dels materials – tot necessita ajust. Aquesta es la part mes iterativa del projecte. Reserva temps extra aqui.

El PDF no es veura professional immediatament. La generacio de PDF es notoriament capritxosa. L'espaiat anira malament. El logotip sera de la mida incorrecta. La taula no s'alineara. Itera. Cada correccio es especifica i rapida.

El Calendari del Cap de Setmana

Aqui tens un calendari realista:

Dissabte al mati: Tasques 1 i 2. Configura el projecte i construeix el formulari. A l'hora de dinar, pots introduir dades de pressupost i desar-les.

Dissabte a la tarda: Tasca 3. La previsualitzacio 3D. Aixo requereix mes iteracio. A l'hora de sopar, tens una paret 3D giratoria amb una finestra que s'actualitza quan canvies el formulari.

Diumenge al mati: Tasca 4. Generacio de PDF. A mig mati, pots generar un document de pressupost professional.

Diumenge a la tarda: Tasques 5 i 6. Historial de pressupostos i poliment. Al vespre, l'aplicacio es veu i funciona com un producte real.

Set tasques. Dos dies. Una aplicacio que fa que el teu oncle sembli una empresa deu vegades mes gran.

La Llico Mes Gran

Mai has escrit una linia de codi. Pero has pres totes les decisions que importaven. Has decidit el model de dades. Has decidit el flux d'usuari. Has decidit com havia de ser la previsualitzacio 3D. Has decidit la formula de preus. Has decidit el disseny visual.

L'agent ha escrit el codi. Tu has fet el pensament. I el pensament – la comprensio del domini, el gust per allo que es veu professional, el criteri sobre el que necessita l'usuari – aixo es la part que cap agent pot fer.

Aixo es el que significa ser un membre de la tripulacio que dirigeix. No un passatger. No un capita. Algu que coneix les aigues, veu els corrents, i diu a la tripulacio cap a on navegar.

Quan les coses surten malament

Figura 13: *Els errors són inevitables. La recuperació és una habilitat.*

Tothom qui treballa amb agents prou temps acaba acumulant una col · lecció d'aquestes històries. Aquells moments en què t'enrecolzes a la cadira, mires la pantalla fixament i dius alguna cosa que la teva mare no aprovaria. Són humiliants. Són educatives. I són inevitables.

Aquest capítol és una col · lecció de coses que realment surten malament quan encomanes feina real a un agent. No són hipòtesis. No són fantasies catastrofistes. Són el tipus d'errors que et costen

una tarda o la teva confiança. Cadascun porta una lliçó que connecta amb alguna cosa dels capítols anteriors d'aquest llibre.

Llegeix-les abans de cometre els mateixos errors. O llegeix-les després, i sent-te menys sol.

El correu que no estava a punt

Vas demanar a l'agent que redactés un correu de seguiment per a un client. L'agent va produir un esborrany perfectament raonable. Li vas fer un cop d'ull ràpid, vas pensar "té bona pinta" i vas prémer enviar.

Dues hores més tard, el client va trucar. El correu feia referència a una reunió que encara no havia tingut lloc — l'agent havia confós l'esdeveniment del calendari del dimarts vinent amb les notes del dimarts passat. També citava un preu d'una proposta antiga, no de l'actual. El correu era coherent, professional, i incorrecte en dos punts que importaven.

La lliçó: No enviïs mai comunicacions generades per un agent sense llegir-les com si les hagués escrit una persona. L'agent no sap què és actual i què està desfasat. No sap que la reunió encara no ha tingut lloc. Va muntar contingut que sonava plausible a partir del context que tenia, i part d'aquell context estava obsolet. Això és el capítol de verificació en acció — i la raó per la qual el gradient de confiança situa "enviar correus" alt a l'escala de revisió.

L'estadística segura de si mateixa

Vas demanar a l'agent que preparés una anàlisi competitiva. Va produir un informe preciós: mides de mercat, taxes de creixement,

xifres d'ingressos de competidors, percentatges de quota de mercat. Impressionant. Detallat. Vas utilitzar tres d'aquells números en una presentació per al consell.

Durant la ronda de preguntes, un membre del consell va obrir l'informe sectorial real que l'agent semblava citar. Dos dels tres números eren incorrectes. No estaven esbojarradament malament — prou propers per ser plausibles, prou diferents per ser vergonyosos. L'agent no havia *trobat* aquells números. Els havia *generat* — xifres plausibles que encaixaven amb la narrativa, presentades amb la mateixa confiança que l'únic número que sí que era correcte.

La lliçó: Aquest és el problema de l'al · lucinació del Capítol 9, manifestant-se en l'escenari de màxim risc possible. Els números dels agents s'han de verificar sempre amb les dades originals. Sempre. Si l'agent cita un informe, busca l'informe i comprova la citació. Si l'agent produeix un percentatge, rastreja'l fins a les dades. L'agent no sap la diferència entre un fet que ha trobat i un fet que s'ha inventat.

El redisseny excessiu

Vas demanar a l'agent que canviés el color d'un botó del teu web de blau a verd. Una tasca ràpida. Cinc minuts.

L'agent va canviar el color del botó. També va notar que l'estil del botó era "inconsistent amb les pràctiques de disseny modernes" i el va actualitzar. Després va actualitzar els altres botons perquè coincidissin. Després la barra de navegació. Després el peu de pàgina. Després va reestructurar el CSS per fer servir un "enfocament més mantenible".

El botó era verd. Tot el resta era irreconeixible. Vuitanta-set fitxers havien canviat. L'agent havia fet un redisseny complet que no li havies demanat, i desfer-ho significava esbrinar quin dels vuitanta-set fitxers contenia l'únic canvi que realment volies.

La lliçó: Restriccions. "Canvia el color del botó de la pàgina d'inici de blau (hex 3b82f6) a verd (hex 22c55e). No canviïs res més." Això és tot el que cal. Els agents són optimitzadors entusiastes — veuen oportunitats de millora i les aprofiten tret que els diguis explícitament que no ho facin. El capítol sobre instruccions existeix exactament per aquesta raó. A més: branques de Git. Si l'agent hagués estat treballant en una branca, l'eliminaries i començaries de nou. Cinc segons. Cap dany.

Les dades que no tenien còpia de seguretat

Vas construir una petita aplicació amb un agent. Funcionava genial. Després vas demanar a l'agent que "netegés la base de dades" — volies dir que eliminés algunes dades de prova que havies introduït mentre experimentaves.

L'agent va interpretar "netejar" com "restablir a un estat net". Va eliminar totes les taules i les va recrear buides. Sis setmanes de dades reals de clients — pressupostos, mesures, informació de contacte — perdudes.

La lliçó: Dues lliçons, en realitat. Primera: sigues terroríficament específic quan la tasca implica dades. "Elimina tots els pressupostos on el nom del client contingui 'test' o 'asdf'" és molt diferent de "neteja la base de dades". Segona: còpies de seguretat. Si estàs construint alguna cosa real — alguna cosa amb dades que importen — les còpies de seguretat regulars de la base de dades no són opcionals. Demana al teu agent que les configuri. És una

de les primeres coses que hauries de fer quan l'aplicació comença a contenir dades reals.

La funcionalitat que ningú volia

Vas demanar a l'agent que afegís un mode fosc a l'aplicació de pressupostos. L'agent ho va fer magníficament. Interruptor a la capçalera, transició suau, tots els colors adaptats. Quedava genial.

El teu oncle ho va odiar. "Per què hi ha un interruptor a la meva capçalera? Jo faig servir això a les obres amb llum del dia. El mode fosc és inútil. I l'interruptor va confondre el meu nou treballador — va pensar que era un botó d'encendre/apagar de tota l'aplicació."

Havies passat dues hores en una funcionalitat basada en la teva pròpia preferència, no en les necessitats del teu usuari. L'agent va executar perfectament. El problema era la instrucció, no l'execució.

La lliçó: Un agent construirà exactament el que li demanis. No et dirà si hauries d'estar-ho demanant. La pregunta "hauríem de construir això?" sempre és teva. Parla amb l'usuari real abans de construir. El teu oncle t'hauria dit en deu segons que el mode fosc era inútil per al seu flux de treball — i potser t'hauria mencionat tres coses que realment necessita.

El patró de recuperació

Totes les històries de guerra anteriors tenen el mateix patró de recuperació:

1. **Atura.** No deixis que l'agent continuï. Si va en la direcció equivocada, més feina ho empitjora.

2. **Avalua.** Què ha passat realment? Què ha canviat? Què s'ha perdut?
3. **Restaura.** Branca de Git? Elimina-la. Base de dades? Restaura des de la còpia de seguretat. Correu? Envia una correcció.
4. **Aprèn.** Quina instrucció o barrera de seguretat hauria previngut això? Afegeix-la al teu procés.
5. **Torna-ho a provar.** Amb millors instruccions, millors restriccions i millor verificació.

Els errors són inevitables. La recuperació és una habilitat. I la prevenció — instruccions clares, restriccions adequades, verificació, còpies de seguretat — és el que tot aquest llibre t'està ensenyant.

Quan fer-ho tu mateix

Figura 14: *De vegades l'eina correcta són les teves pròpies mans.*

L'habilitat més important per treballar amb agents no és saber com utilitzar-los. És saber quan *no* fer-ho.

Un agent és una eina potent. Una serra circular talla més ràpid que una serra de mà. Però no fas servir una serra circular per retallar un bonsai. Algunes feines requereixen un toc humà — no perquè l'eina sigui dolenta, sinó perquè la feina demana alguna cosa que l'eina no pot proporcionar.

La zona del criteri

Hi ha tasques on la velocitat de l'agent és el que importa i tasques on en realitat és el problema. La distinció normalment es redueix a una paraula: *criteri*.

Un agent pot redactar un correu. Però hauria de redactar el correu que diu a un client de tota la vida que puges els preus? Probablement no — almenys no sense una edició a fons. El to, la relació, l'equilibri delicat entre honestedat i diplomàcia — això és territori humà.

Un agent pot analitzar dades. Però hauria de decidir si acomiadar un empleat basant-se en aquesta anàlisi? Evidentment no. Les dades informen la decisió. La decisió és teva.

Un agent pot escriure un informe. Però hauria d'escriure l'avaluació de rendiment d'algú del teu equip? Les paraules potser són correctes. El *pes* d'aquelles paraules — que venen d'una eina en lloc d'una persona — ho canvia tot.

El patró: els agents són excel · lents *produint* coses. No són fiables *decidint* coses que afecten les persones. Utilitza'ls per al primer. Reserva't el segon.

Quan el context ho és tot

Algunes tasques requereixen un context que no cap en un prompt. La història d'una relació. Les dinàmiques polítiques d'un equip. La raó no dita per la qual es va prendre una decisió fa tres anys. El fet que el CEO del teu client odia les llistes amb vinyetes.

Un agent treballa amb el que li dones. Si la informació més important és del tipus que no pots articular fàcilment — sentiments,

instints, relacions — l'agent se la perdrà. I no sabrà que se l'està perdent.

Aquestes són les tasques on la teva experiència, el teu criteri i la teva humanitat són precisament el que importa. Cap prompt pot capturar "fa set anys que treballo amb aquest client i sé exactament com reaccionarà a això". Aquest coneixement és teu. Fes-lo servir.

Quan la velocitat és l'enemiga

De vegades el valor d'una tasca *és* el temps que porta. Escriure una nota d'agraïment sincera. Pensar a fons una decisió estratègica. Esbossar una idea al paper per veure si queda bé. Llegir un document lentament per entendre'n els matisos.

Un agent pot fer totes aquestes coses més ràpid. Però més ràpid no sempre és millor. Alguns pensaments necessiten passar a velocitat humana — aquell tipus de processament lent i deliberat on les idees sorgeixen de l'acte de fer, no del resultat.

Si et trobes que fas servir un agent per a tot i et sents estranyament desconnectat de la teva pròpia feina — aquesta és la senyal. Fes un pas enrere. Fes alguna cosa a mà. La pèrdua d'eficiència val la pena per la comprensió que guanyes.

Quan el risc és massa alt

Hi ha àmbits on el cost d'un error és catastròfic i la capacitat de verificació és limitada:

- **Documents legals** on una clàusula incorrecta crea responsabilitat
- **Informació mèdica** on un fet incorrecte crea perill

- **Càlculs financers** on un número incorrecte crea pèrdues
- **Configuracions de seguretat** on un paràmetre incorrecte crea exposició

Un agent pot ajudar en tots aquests àmbits. Pot redactar, calcular, suggerir i verificar. Però la revisió final, l'aprovació final, el "sí, això és correcte i jo en sóc responsable" — això ha de ser humà. Concretament, un humà amb l'experiència rellevant.

Fer servir un agent per redactar un contracte està bé. Enviar aquell contracte sense que un advocat el revisi, no. L'agent no sap el que no sap. Tampoc ho sap un no-expert que revisi el seu resultat.

El marc de decisió

Quan estiguis decidint si fer servir un agent per a una tasca, feste tres preguntes:

1. **Puc verificar el resultat?** Si la resposta és sí, probablement l'agent és segur de fer servir. Si no pots saber si el resultat és correcte — perquè et falta l'experiència, o la verificació portaria tant de temps com fer-ho tu mateix — pensa-t'ho dues vegades.
2. **Què passa si és incorrecte?** Si un error és fàcil de detectar i barat de corregir, deixa que l'agent treballi. Si un error és difícil de detectar i car de recuperar, mantingues un humà en el procés — o fes-ho tu mateix.
3. **L'element humà és el que importa?** Si el valor de la tasca ve del fet que la facis *tu* — la relació, el criteri, l'expressió creativa — llavors fer-ho més ràpid amb un agent en realitat destrueix valor.

Un agent és una eina. Com totes les eines mai inventades, el mestre no és qui la fa servir més — sinó qui sap exactament quan deixar-la de banda.

Ser l'humà en el procés

Figura 15: *Tu ets el control de qualitat.*

Hi ha una expressió que sentiràs cada vegada més en els propers anys: "human in the loop" — l'humà en el procés. Significa exactament el que sembla — un sistema on una IA fa la feina, però un humà revisa, aprova o redirigeix en els punts clau.

Tu ets aquell humà.

Aquest capítol tracta sobre com és realment aquest paper — no en teoria, sinó en la pràctica diària. Què significa ser la persona que comprova la feina de l'agent, detecta els seus errors i aporta el criteri que ell no pot tenir.

No et substitueixen

Traiem-nos això de sobre, perquè és l'ansietat que hi ha darrere de tot.

Els agents cada vegada són millors. Escriuen codi, redacten informes, analitzen dades, generen dissenys. És natural mirar-ho i pensar: "Què em queda a mi?"

Tot el que importa.

Un agent pot produir un informe financer. Tu saps si els números expliquen la història correcta. Un agent pot redactar un correu per a un client. Tu saps si el to serà l'adequat. Un agent pot dissenyar una interfície d'usuari. Tu saps si tindrà sentit per a la persona que la farà servir a les 7 del matí en una obra amb guants posats i zero paciència.

Coneixement del sector, gust, criteri, relacions, empatia, context, política, timing — aquestes no són funcionalitats que puguis afegir a un model de llenguatge. Són les coses que fan que la feina *funcioni*. Són el que tu aportes.

L'agent multiplica la teva producció. No substitueix la teva aportació.

El ritme de revisió

A la pràctica, ser l'humà en el procés significa establir un ritme:

Per a tasques curtes (minuts): L'agent produeix un resultat, tu el revises immediatament. Redacta un correu — llegeix-lo, ajusta'l, envia'l. Genera un gràfic — comprova les dades, retoca les etiquetes, exporta'l. Cicle ràpid.

Per a tasques mitjanes (hores): L'agent treballa per etapes, i tu revises entre etapa i etapa. Construeix el formulari — revisa. Afegeix la previsualització 3D — revisa. Genera el PDF — revisa. Cada punt de control és una oportunitat per reconduir abans que l'agent avanci més en la direcció equivocada.

Per a tasques llargues (dies): L'agent treballa en branques, i tu revises pull requests. Aquest és el flux de treball de Git del Capítol 3. L'agent proposa canvis, tu revises els canvis, i tu decideixes què es fusiona. Si alguna cosa està malament, ho rebutges i proporciones millors instruccions.

El ritme s'adapta al risc. La feina de baix risc rep un cop d'ull ràpid. La feina d'alt risc rep una revisió acurada. Però el ritme no s'atura mai — perquè el moment en què deixes de comprovar és el moment en què l'agent es desvia.

Què estàs comprovant realment

Quan revises el resultat de l'agent, no estàs comprovant si el codi compila o si la gramàtica és correcta. D'això se n'encarrega l'agent. Estàs comprovant coses que l'agent *no pot* comprovar:

Resol el problema correcte? L'agent resol el problema que tu has descrit. Tu estàs comprovant si el problema que has descrit és el problema que realment existeix.

Encaixa amb el context? L'agent no coneix la història del teu client, la cultura del teu equip, les normes no escrites de la teva empresa. Tu sí. El resultat encaixa amb aquest context?

És apropiat? Apropiat en to, en abast, en ambició. L'agent no sap que la prioritat d'aquest trimestre és l'estabilitat, no les noves funcionalitats. No sap que aquest client aprecia la brevetat. No sap que el teu cap és escèptic amb el contingut generat per IA.

Hi posaries el teu nom? La prova definitiva. Si aquest resultat surt amb el teu nom — el teu pressupost, el teu informe, el teu correu — t'hi sents còmode? Si no, no està acabat.

La bretxa del gust

Hi ha un concepte en el treball creatiu anomenat la "bretxa del gust" — la distància entre el que pots reconèixer com a bo i el que pots produir. Quan comences en qualsevol camp, el teu gust supera la teva capacitat. Saps quin aspecte té el bon, però encara no el pots crear.

Els agents redueixen aquesta bretxa dràsticament. El teu gust — el teu sentit del que és correcte, del que és professional, del que funciona — és la guia. L'agent proporciona la velocitat d'execució. Tu proporciones la direcció.

Per això algú com en Morten, l'instal·lador de finestres, va poder construir una aplicació de pressupostos d'aspecte professional en un cap de setmana. Ha passat quinze anys desenvolupant *gust* per com ha de ser un bon pressupost, quina informació necessita un client, com ha de ser una interacció professional. No sabia escriure el codi. Però sabia reconèixer si el resultat era correcte — i reconduir-lo fins que ho fos.

El teu gust és el teu superpoder. L'agent és només l'amplificador.

Quan anul·lar la decisió de l'agent

De vegades l'agent fa alguna cosa que és tècnicament correcta però equivocada per raons que no pot entendre. Aquests són els moments que defineixen l'humà en el procés:

- L'informe és precís però el to no és l'adequat per a aquesta audiència.
- La funcionalitat funciona però el flux d'usuari és confús.
- L'anàlisi és sòlida però la conclusió no té en compte el context polític.
- El correu és correcte però ha de ser més càlid — aquest client acaba de perdre un familiar.

En cada cas, l'agent no pot saber el que tu saps. Anul·la-ho. Canvia-ho. Aquests són els moments en què et guanyes el teu lloc en el procés — no perquè l'agent estigui espatllat, sinó perquè algunes coses requereixen un humà.

La visió a llarg termini

Ser l'humà en el procés no és un paper temporal fins que els agents millorin. És el paper *permanent*. Les tasques concretes canviaran. Les eines milloraran. Però la necessitat de criteri humà, gust humà i responsabilitat humana no anirà enlloc.

La pregunta no és si estaràs en el procés. És si en seràs *bo*. I ser-ne bo significa desenvolupar exactament les habilitats que aquest llibre ensenya: comunicació clara, escepticisme sa, experiència en el teu àmbit, i la confiança per anul·lar una màquina quan el teu criteri diu el contrari.

No ets la persona que escriu. Ets la persona que decideix. Aquesta és la feina més important de qualsevol tripulació.

Parlant amb el Teu Equip Tècnic

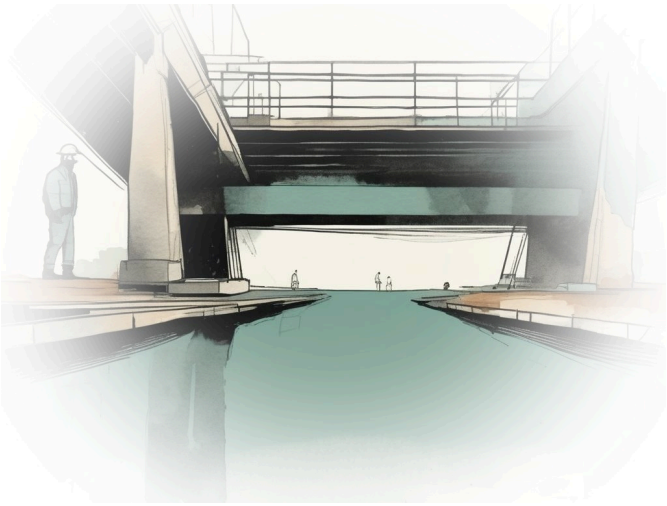

Figura 16: *Ara parles dos idiomes.*

Acabes de passar tretze capítols aprenent com funciona el programari, què són els agents i com dirigir-los. Ara tens una cosa que la majoria de persones no tècniques no tenen: un vocabulari compartit amb els teus desenvolupadors.

Aquest capítol tracta de com fer-lo servir. No per fer veure que ets enginyer — sinó per tancar l'escletxa entre "no entenc què esteu construint" i "entenc prou per ajudar-vos a construir la cosa correcta."

El Vocabulari

Aquí tens una referència dels termes que has après, emmarcats tal com apareixen en converses reals:

Quan el teu desenvolupador diu **"l'API està retornant un 500"** — la cuina (Django) s'ha espatllat. Alguna cosa ha explotat al costat del servidor. El menjador (React) funciona bé. El problema és entre bastidors.

Quan diuen **"hem d'escriure una migració"** — l'estructura de la base de dades (Postgres) ha de canviar per donar suport a una nova funcionalitat. Com afegir un calaix nou a l'arxivador.

Quan diuen **"està en memòria cau a Redis"** — les dades s'estan servint des de la pissarra ràpida en lloc de l'arxivador gran. Si les dades semblen desfasades, la pissarra potser necessita actualitzar-se.

Quan diuen **"ho posaré en una branca"** — estan treballant en un univers paral · lel (branca de Git) perquè l'aplicació principal no es vegi afectada. L'experiment és segur.

Quan diuen **"el PR necessita revisió"** — hi ha un pull request a punt. Algú ha proposat canvis i vol un segon parell d'ulls abans de fusionar-los.

Quan diuen **"el CI està fallant"** — la llista de verificació automatitzada ha detectat un problema. El canvi ha trencat un test o ha violat una regla. Cal arreglar-ho abans de poder-ho publicar.

Quan diuen **"el DNS no s'ha propagat"** — la guia telefònica d'internet encara no s'ha actualitzat a tot arreu. Algunes persones poden veure el canvi, d'altres no. Espera.

Quan diuen **"hauríem d'afegir un WebSocket per a això"** — en lloc que la pàgina comprovi si hi ha actualitzacions refrescant-

se, volen afegir una connexió en directe perquè les actualitzacions apareguin instantàniament.

Quan diuen **"la finestra de context"** — el banc de treball de l'agent. Quanta informació pot retenir alhora.

Quan diuen **"ha al·lucinat"** — l'agent s'ha inventat alguna cosa. Amb confiança. Sembla real però no ho és.

Preguntes que Val la Pena Fer

La cosa més valuosa que pots fer en una conversa tècnica no és entendre cada detall. És fer les preguntes correctes. Aquí en tens algunes que et faran guanyar respecte:

"Què veu realment l'usuari quan passa això?" — Això ancora qualsevol discussió tècnica a la realitat. Els desenvolupadors de vegades es perden en detalls d'implementació. Aquesta pregunta els fa tornar a l'experiència.

"Quin és el risc si això surt malament?" — Això és el gradient de confiança aplicat a la conversa. Separa les coses que val la pena preocupar-se de les que sonen espantoses però no ho són.

"Podem provar-ho primer en una branca?" — Ja saps què és una branca. Suggerir això demostra que entens que els experiments poden ser segurs.

"Això és alguna cosa que un agent podria gestionar?" — No per substituir el teu desenvolupador — sinó per suggerir que potser l'script de migració tediós o la tasca d'estilització repetitiva es podrien delegar. Els desenvolupadors de vegades s'obliden d'utilitzar les seves pròpies eines.

"Què faria això més fàcil de testejar?" — Això demostra que entens que la verificació importa. És una de les preguntes més fluides en enginyeria que pot fer algú que no és enginyer.

"Com és el model de dades?" — Ja saps què és una base de dades. Saps de taules i files. Preguntar pel model de dades demostra que estàs pensant en l'estructura, no només en les funcionalitats.

Ser Útil en la Planificació

La contribució més gran que pots fer a un equip tècnic no és escriure codi ni revisar pull requests. És *definir el problema correctament*.

Els enginyers estan optimitzats per construir solucions. Sovint no són tan bons qüestionant si és la solució correcta. Aquí és on entres tu — especialment si estàs més a prop dels usuaris:

"Estem construint un sistema de notificacions" — pots preguntar: "Hem parlat amb els usuaris sobre si volen notificacions? De quin tipus? Amb quina freqüència? Les últimes tres aplicacions que he fet servir tenen un problema de 'fatiga de notificacions'."

"Estem afegint un mode fosc" — pots preguntar: "Qui ho ha demanat? Els nostres usuaris són instal · ladors de finestres en obres amb molta llum. Potser hauríem de fer un mode clar d'alt contrast."

"L'agent gestionarà l'onboarding de clients" — pots preguntar: "Què passa quan l'agent s'equivoca? Hi ha un traspàs a un humà? Com sabrà el client que està parlant amb un agent?"

Aquestes no són preguntes tècniques. Són preguntes de *producte*. I sovint són més valuoses que qualsevol decisió tècnica que es prengui a la mateixa reunió.

El Pont

Ets un pont. Ara parles dues llengües — no amb fluïdesa en la tècnica, però prou bé per traduir entre les persones que construeixen el producte i les persones que el fan servir.

Aquest pont no existia abans que llegissis aquest llibre. Els desenvolupadors parlaven en APIs i migracions i pipelines de CI. Els usuaris parlaven en frustracions i desitjos i "estaria bé si." L'escletxa entre aquestes dues converses és on els productes fracassen.

Ara estàs al mig d'aquesta escletxa. No com a enginyer. No com a usuari. Com algú que entén prou d'ambdós costats per assegurar-se que estan construint la mateixa cosa.

Això no és un paper petit. És el paper que determina si el vaixell arriba a l'illa correcta.

Mantenir el Pols

Figura 17: *Deu minuts, tres cops per setmana.*

La tecnologia avança ràpid. Les eines d'aquest llibre evolucionaran. Apareixeran nous agents. Es llançaran models millors. Els noms concrets — Claude, React, Django — podrien ser diferents d'aquí a tres anys.

Però els *patrons* no canviaran. Els agents seguiran necessitant instruccions clares. El programari seguirà tenint frontends i backends. La verificació seguirà importat. El que canvia és el paisatge al voltant d'aquests patrons — i mantenir-se al dia amb aquest paisatge és una habilitat que val la pena desenvolupar.

No cal que et converteixis en un addicte a les notícies tecnològiques. Necessites deu minuts, tres vegades per setmana.

Els Tres Punts de Trobada

Hi ha tres llocs on el món tecnològic es congrega per compartir què hi ha de nou, què funciona i què s'ha trencat. Cadascun té un sabor diferent.

Hacker News
URL: news.ycombinator.com

Gestionat per Y Combinator — la incubadora de startups darrere d'Airbnb, Dropbox, Stripe i centenars d'altres. Enginyers, fundadors i investigadors publiquen enllaços i els discuteixen. La portada és una llista curada del que el món tecnològic està mirant *avui*.

Per què t'importa: Les seccions de comentaris. Veuràs enginyers reals discutint si una eina nova és realment bona o només soroll mediàtic. Algú llança un nou framework d'agents d'IA — el primer comentari explica què hi ha de genuïnament nou i el segon comentari explica per què fracassarà. Aquest tipus de discussió honesta i experta és gairebé impossible de trobar en cap altre lloc.

Com fer-lo servir: Fes un cop d'ull a la portada. Llegeix els títols. Clica en un o dos que t'interessin. Llegeix els tres primers comentaris. Cinc minuts. Aprendràs més sobre què pensa la indústria tecnològica que amb qualsevol butlletí informatiu.

GitHub Trending
URL: github.com/trending

Una llista diària i setmanal de projectes de codi obert que estan guanyant tracció. Són eines i llibreries reals que els desenvolupadors estan destacant i clonant — un senyal en temps real del que s'està fent popular.

Per què t'importa: No entendràs el codi. No cal. Llegeix les descripcions dels projectes i els READMEs. "Una alternativa autoallotjada a Notion." "Framework d'agents d'IA per construir fluxos de treball autònoms." "Client ràpid de Postgres per a Python." Comences a reconèixer patrons: quins problemes es resolen una vegada i una altra, quines categories segueixen sent tendència, què podria estar avaluant el teu equip el trimestre vinent.

Com fer-lo servir: Comprova-ho un cop per setmana. Mira els deu primers. Llegeix les descripcions d'una línia. Si alguna cosa sembla rellevant per al teu treball o el teu equip, desa-la i esmenta-la la pròxima vegada que siguis en una reunió de planificació. "He vist un projecte a GitHub Trending que fa exactament el que hem estat parlant."

Reddit
Subreddits: r/programming, r/webdev, r/selfhosted, r/artificial, r/LocalLLaMA

Més informal que Hacker News, més opinat i sovint més pràctic. La gent comparteix el que realment està construint, el que realment s'ha trencat i el que realment pensa sobre les últimes eines — sense el vernís professional.

Per què t'importa: És la visió "a peu de carrer" de la tecnologia. Quan surt un nou model d'IA, Reddit et diu en qüestió d'hores si és genuïnament millor o només màrqueting. Quan una eina té un defecte crític, Reddit el treu a la llum abans que la premsa tecnològica. I r/selfhosted és una mina d'or per descobrir eines que pots executar tu mateix.

Com fer-lo servir: Subscriu-te a dos o tres subreddits rellevants. Fes un cop d'ull quan tinguis uns minuts. La proporció de senyal respecte al soroll és més baixa que a Hacker News, però les idees pràctiques sovint són més agudes.

L'Efecte Compost de la Consciència

No es tracta de convertir-se en un expert en tecnologia. Es tracta de reconeixement de patrons.

Després d'un mes de lectura casual, començaràs a reconèixer noms. "Oh, aquest és el framework que el meu equip estava avaluant." "Aquesta és l'empresa que fa l'agent que faig servir." "He vist tres projectes aquesta setmana fent el mateix — això deu ser una tendència."

Després de tres mesos, desenvoluparàs opinions. "El nostre equip hauria de mirar això." "Aquesta aproximació sembla sobredimensionada." "Això resol un problema que realment tenim."

Després de sis mesos, seràs la persona no tècnica de la sala que fa que les persones tècniques s'aturin i escoltin. No perquè sàpigues programar. Perquè has estat parant atenció.

Aquesta consciència — la comprensió del que està passant, del que ve i del que importa — és la diferència entre algú que utilitza la tecnologia i algú que la *navega*. I navegar-la és exactament el que fa un bon membre de la tripulació.

Construir el Teu Filtre

Ràpidament notaràs que la majoria de notícies tecnològiques cauen en unes quantes categories:

Senyal: Eines o aproximacions noves que resolen problemes reals. Val la pena llegir-les.

Soroll mediàtic: Màrqueting disfressat de notícies. Una empresa ha llançat alguna cosa i vol que t'emocionis. Fes un cop d'ull als comentaris de Reddit per veure si l'entusiasme és real.

Drama: Política d'empresa, controvèrsies de CEO, guerres culturals. Entretingut però rarament útil.

Immersions profundes: Articles llargs i tècnics sobre problemes específics. Salta'ls tret que el tema sigui directament rellevant per a tu.

Aprendre a separar el senyal del soroll és una habilitat. Comença ampli, després estreny. En poques setmanes, sabràs quines fonts ofereixen senyal de manera consistent i quines són principalment soroll. Segueix el senyal. Ignora la resta.

Deu minuts, tres vegades per setmana. Això és tot el que cal per anar un pas per davant.

Paraules Finals

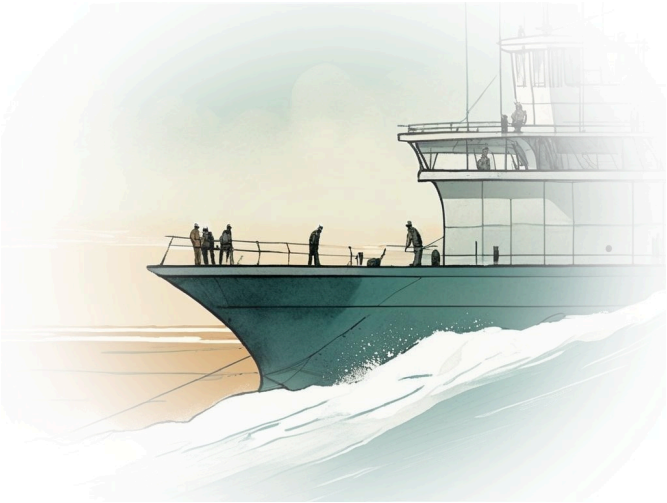

Figura 18: *Ara vés i construeix alguna cosa.*

Has començat aquest llibre com algú que se'n sortia bé amb els ordinadors. L'acabes com algú que entén com es construeix el programari, com funcionen els agents d'IA i com dirigir-los per construir coses reals.

Això no és un canvi petit.

Fa un any, la idea de construir una aplicació funcional en un cap de setmana — amb una previsualització 3D, una base de dades, un generador de PDF i una interfície professional — hauria semblat absurda sense un equip de desenvolupament. Ara és un projecte de cap de setmana. No perquè la tecnologia sigui màgia, sinó perquè has après l'habilitat que la desbloqueja: la capacitat de pensar amb claredat sobre el que vols i comunicar-ho amb precisió.

Què Has Après

Saps què hi ha sota el capó. Un frontend (React) que l'usuari veu. Un backend (Django) que fa complir les regles. Una base de dades (Postgres) que ho recorda tot. Una memòria cau (Redis) que manté les coses ràpides. Una API que ho connecta tot. Git que manté tot segur. DNS que ho fa localitzable. CI que assegura que res no es trenqui.

Saps què són els agents. No són màgia. No són conscients. Un bucle: observar, planificar, actuar, comprovar, repetir. Poderosos quan es dirigeixen bé. Perillosos quan es deixen sense supervisió.

Saps com dirigir-los. Instruccions clares amb restriccions. Materials en brut en lloc de descripcions vagues. Verificació en lloc de confiança cega. El gradient de confiança — estret al principi, relaxat amb evidència.

I saps quan intervenir. Quan el judici importa més que la velocitat. Quan l'element humà és el que compta. Quan hi ha massa en joc per a suposicions fetes amb confiança.

La Metàfora de la Tripulació

Aquest llibre s'anomena la Guia del Membre de la Tripulació per una raó. A The Agentic Crew, hi ha capitans — els enginyers que construeixen el vaixell i coneixen cada post. Hi ha cadets — els nens que tot just estan aprenent què és un vaixell. I hi ha membres de la tripulació — persones amb habilitats, coneixements i rols que no impliquen construir el vaixell però que absolutament impliquen navegar-lo.

Tu ets tripulació. I un vaixell sense una bona tripulació no va enlloc.

El capità potser sap com construir el motor, però tu coneixes les aigües. Coneixes els clients. Coneixes el negoci. Saps com és l'èxit des de coberta, no només des del timó. Aquesta perspectiva no és secundària. És essencial.

Què Ve Després

Les eines canviaran. Els models milloraran. Apareixeran nous agents que faran que les eines d'avui semblin primitives. No passa res. Els principis d'aquest llibre — comunicació clara, escepticisme saludable, coneixement del domini i la voluntat de prendre el timó — no estan lligats a cap eina específica. Estan lligats a com els humans treballen amb sistemes poderosos. I això no canvia.

El que espero que t'emportis d'aquest llibre no és un conjunt de tècniques. És una *confiança*. La confiança que pots seure davant d'un agent d'IA i construir alguna cosa real. La confiança que les teves idees, el teu coneixement del domini i el teu criteri són exactament el que es necessita. La confiança que "no soc tècnic" mai va ser realment cert — simplement no havies trobat les eines adequades.

Les has trobades.

Ara ves i construeix alguna cosa.

A tots els qui els van dir
"no ets prou tecnic."
Ho ets. Sempre ho has estat.
Ara les eines hi estan d'acord.