

O PLATFORMĂ • TREI PRODUSE

AI pentru hosting, aplicații și comerț.

Sysadmin cPanel

App Builder

WooCommerce

Codex CLI / Codex SDK TypeScript / OpenCode

Chat comun

Procese la cerere

Izolare per client

Arhitectură, integrare cPanel, permisiuni, Abilities,
publicare, contracte interne și criterii de acceptare.

VERSIUNEA 2.0 • 16 SEPTEMBRIE 2026

Propunere pentru review tehnic. Configurațiile nu au fost testate
pe infrastructura HostGate. Înlocuiește și extinde studiul WooCommerce v1.0.

Cuprins

Document unificat pentru review. Capabilitățile documentate sunt citate; componentele HostGate și fluxurile de lucru sunt propuneri de implementare.

01. Decizia executivă	3
02. Context verificat, cerințe și necunoscute	4
03. O platformă, trei profiluri, un chat	5
04. Arhitectura comună și granițele de încredere	6
05. Pluginul cPanel: interfață, autentificare, integrare	7
06. Profiluri de execuție și read-only real	8
07. Sysadmin pentru contul cPanel	10
08. App Builder în același plugin	11
09. WooCommerce: același motor, alt domeniu operațional	13
10. Instrumente locale, Abilities și MCP	15
11. Izolarea și execuția per utilizator	17
12. Varianta A – Codex CLI direct	18
13. Varianta B – Codex SDK TypeScript	19
14. Varianta C – OpenCode CLI per utilizator	21
15. Comparatie și contract de engine	22
16. Joburi, aprobări și operațiuni concurente	23
17. Publicare și verificare independentă	25
18. Date, API intern și componente de construit	26
19. Modele, chei, consum și capacitate	27
20. Securitate, confidențialitate și incidente	28
21. Teste de acceptare și benchmark obiectiv	29
22. Implementare etapizată și alternative	31
23. Decizii pentru review și concluzie	32
Anexa A. Manifest de compatibilitate pentru pilot	33
Anexa B. Checklist operațional per job	33
Anexa C. Surse primare și limite ale verificării	34

01. Decizia executivă

Da: sysadminul și App Builderul pot fi același plugin de chat în cPanel. WooCommerce poate folosi aceeași platformă și chiar aceeași interfață, dar are un modul de integrare și reguli de operare distincte. Nu sunt necesare trei bucle agentice proprii și nici nouă platforme pentru combinațiile dintre trei produse și trei engine-uri.

Propunerea este un singur produs tehnic, cu trei profiluri: **Hosting Assistant**, pentru diagnostic și operațiuni asupra contului; **App Builder**, pentru creare și dezvoltare de aplicații; **Commerce Assistant**, pentru WordPress/WooCommerce. Aceste denumiri sunt propuneri HostGate, nu produse existente ale furnizorilor.

Pluginul din cPanel este intrarea în produs. Agentul efectiv este Codex sau OpenCode, pornit la cerere într-un mediu autorizat. Platforma furnizează instrumentele, limitează resursele, verifică rezultatul și controlează publicarea. cPanel documentează dezvoltarea de pluginuri și interfețe proprii; faptul că interfața este în cPanel nu oferă însă automat izolare sau o integrare agentică gata făcută. [S01, S02]

Răspunsurile care fixează direcția

Întrebare	Decizia propusă
Poate același chat să repare un site și să creeze o aplicație?	Da. Interfață comună, profiluri de execuție și instrumente distincte.
Este nevoie de un agent nou scris de HostGate?	Nu pentru bucla agentică. Se reutilizează agentul existent.
Instalare globală sau per client?	Executabile și worker comune, administrate de HostGate; stare și procese separate pe cont/proiect/profil.
Sunt necesari aproximativ 1.000 de agenți permanenți?	Nu. Procese la cerere, conversații persistente și limită de concurență.
Trebuie MCP?	Nu. WP-CLI/Abilities pentru WordPress; API-uri și tools proprii pentru hosting și aplicații. MCP rămâne o interfață opțională.
Agentul clientului poate deveni root?	Nu în produsul pentru clienți. Operațiunile de server sunt o zonă separată, controlată de HostGate.
Codex SDK este un agent mai puternic decât CLI?	Varianta TypeScript controlează același CLI; alegerea este în primul rând una de integrare. [S11–S13]
OpenCode cere un HTTP comun cu acces la toate conturile?	Nu. Există execuție CLI; comportamentul intern depinde de versiune și se validează. [S20, S25]

Recomandarea inițială

Aș construi o singură fundație **HostGate**, cu un adaptor pentru engine. Aș începe cu Codex CLI într-un backend predominant PHP sau cu SDK-ul TypeScript într-un worker Node.js. Aș integra OpenCode suficient pentru comparație pe aceleași sarcini, fără să îl exclud din cauza unei presupuse obligații de a folosi un server HTTP comun.

Pentru lansare, aş oferi mai întâi diagnostic de cont, reparaţii controlate şi aplicaţii dintr-un catalog restrâns de runtime-uri. WooCommerce ar adăuga pluginul propriu de Abilities şi o temă/blocuri susţinute. Nu aş promite de la început orice aplicaţie, orice plugin şi administrare nelimitată a serverului.

Cea mai importantă decizie nu este CLI versus SDK. Este diferenţa dintre „agent cu accesul complet al clientului” şi „produs care impune tehnic read-only, preview şi aprobare înainte de live”. A doua variantă cere bariere reale, nu doar instrucţiuni în prompt.

02. Context verificat, cerinţe şi necunoscute

Ce păstrăm din discuţii

Contextul recuperat din cealaltă conversaţie confirmă intenţia de a reutiliza un agent existent, de a-l porni sub utilizatorul Linux/cPanel şi de a evita procesele permanente pentru toate conturile. Se păstrează cerinţele de read-only/confirmări, acces la fişiere/API/shell şi comparaţie obiectivă între Codex şi OpenCode. Recuperarea este un rezumat relevant, nu transcriptul integral al discuţiei cu celălalt agent.

Experienţa descrisă de utilizator — un agent folosit prin SSH care investighează un site, execută verificări şi continuă până la o remediere — este modelul funcţional dorit. Documentul nu tratează însă acea experienţă ca dovadă că arhitectura multi-client, izolarea ori publicarea automată sunt deja implementate sau testate.

Cerinţele de bază sunt: instalare comună, execuţie per utilizator, stare separată, procese la cerere, chat propriu, posibilitatea modificării codului şi folosirea unor operaţiuni deterministe unde acestea există. Numărul de aproximativ 1.000 de utilizatori reprezintă scenariul de proiectare menţionat, nu concurenţa măsurată.

Ce nu este confirmat

Nu sunt confirmate versiunile cPanel/CloudLinux, kernelul, integrarea CageFS/LVE, limbajul backendului, configuraţia PHP, runtime-urile disponibile, resursele fizice şi numărul real de lucrări simultane. Nu presupunem un magazin sau o aplicaţie per UID; când există mai multe site-uri în acelaşi cont, izolarea suplimentară devine o decizie explicită.

Prin „App Builder” documentul înţelege creare, editare, testare şi publicare de aplicaţii web găzduite de HostGate, nu doar generare de pagini statice şi nici instalarea arbitrară a unor servicii de sistem. Catalogul de tehnologii suportate trebuie aprobat înainte de lansare.

Cum se citesc afirmaţiile

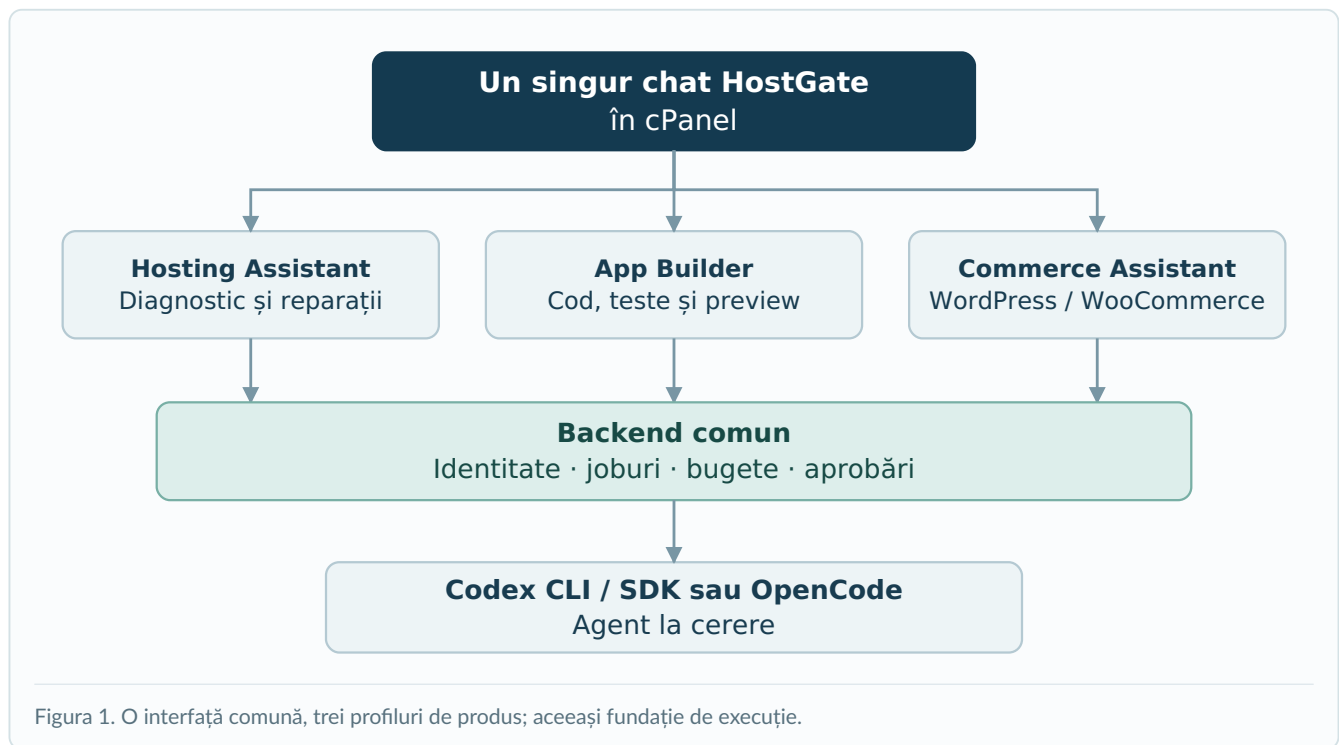
Capabilitate documentată: este susţinută de o sursă primară numerotată, de exemplu existenţa `codex exec --json` sau a `wp ability run`. **Propunere HostGate:** este un mecanism de construit, o regulă ori o structură de date din acest document. **De validat:** depinde de versiunea şi mediul efectiv; nu este prezentată drept funcţie garantată.

Numele `hostgate-tools`, `EngineAdapter`, `ChangeSet`, `ApplicationManifest` şi toate operaţiunile cu prefix `hostgate/` sunt convenţii propuse. Nu sunt comenzi sau API-uri native existente în Codex, OpenCode ori cPanel.

Nu sunt oferite estimări de cost, RAM, viteză sau termene fără măsurători. Sursele care trimit la ramuri `main/dev` descriu codul consultat la data documentului, nu un release fixat pe serverul HostGate.

03. O platformă, trei profiluri, un chat

Schema produsului



Interfața poate avea un selector de site/aplicație și un mod vizibil: **Diagnostic**, **Dezvoltare** sau **Magazin**. Nu este obligatoriu ca utilizatorul să aleagă modul înaintea fiecărei cereri. Agentul poate propune încadrarea, dar backendul verifică profilul și drepturile înaintea pornirii execuției.

Exemplu: clientul scrie „Site-ul nu mai pornește”. Chatul deschide un diagnostic al aplicației selectate. Agentul identifică o eroare în cod și propune o reparație. Platforma pregătește un workspace de dezvoltare, iar clientul vede preview-ul și modificarea. După publicare, aceleași verificări ale profilului sysadmin confirmă că aplicația funcționează.

Același chat nu înseamnă un proces unic cu reuniunea tuturor permisiunilor. Poate exista o conversație vizibilă comună, în spate cu execuții și sesiuni diferite pentru diagnostic, dezvoltare și administrarea magazinului.

Schimbarea profilului

La trecerea într-un profil cu alte drepturi, backendul reevaluează autorizarea și creează sau reia o sesiune corespunzătoare aceluși profil. Transferă un rezumat verificat și referințe la artefacte, nu credențialele ori întregul istoric tehnic. Agentul nu își acordă singur instrumente suplimentare.

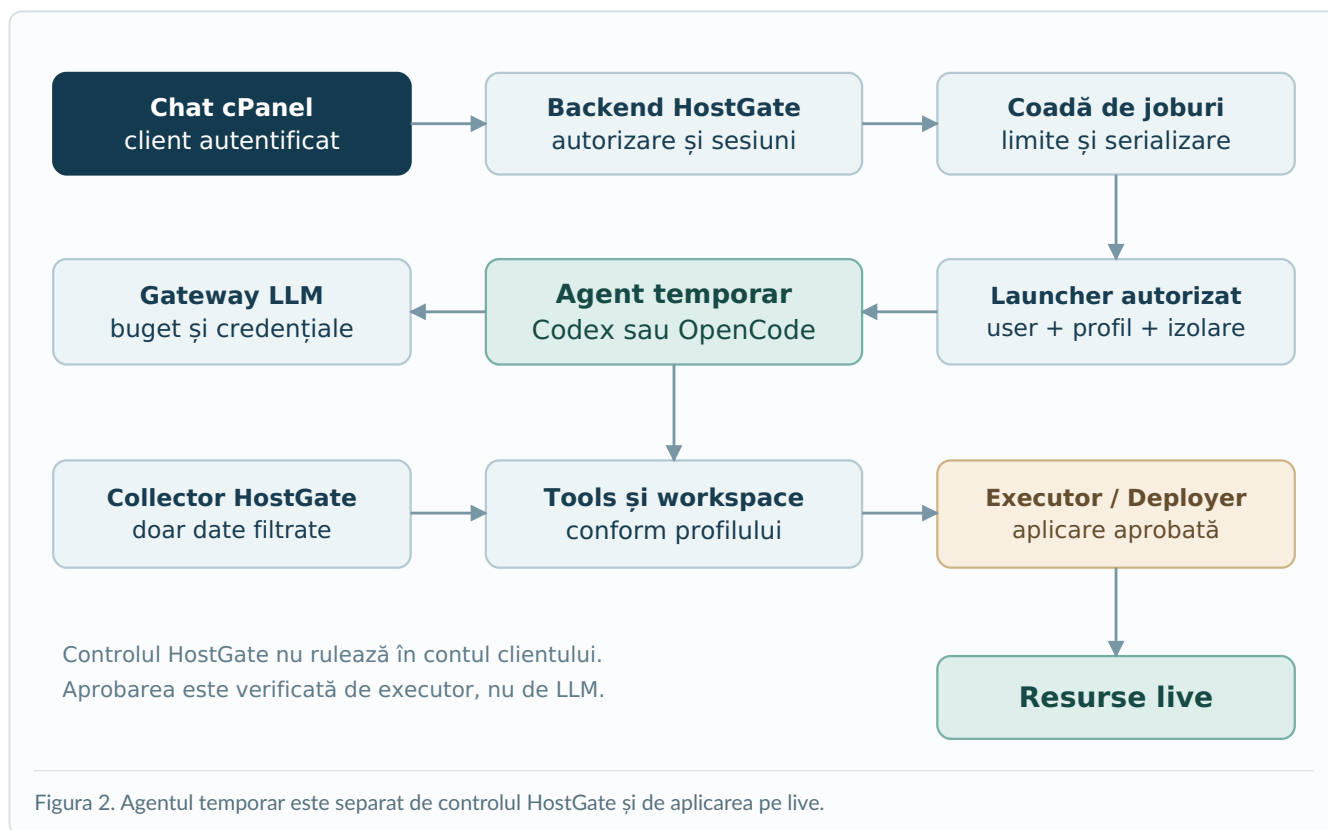
Modulul sysadmin poate fi disponibil și pentru un site care nu a fost creat de App Builder. Invers, fiecare aplicație creată de builder primește un manifest și teste de sănătate pe care sysadminul le poate folosi ulterior. Aceasta este legătura utilă dintre cele două produse, nu necesitatea a doi agenți care se coordonează permanent.

Ce se reutilizează

Comun tuturor	Specific hosting	Specific aplicații	Specific WooCommerce
Chat, sesiuni și joburi	Inventar cont și domenii	Catalog de runtime-uri	Plugin HostGate WordPress
Engine adapter și launcher	Loguri și diagnostic	Workspace și build	Abilities și roluri WP
Bugete și audit	Operațiuni UAPI	Teste și preview	Produse, stoc, comenzi
Aprobări și verificare	Reparații de cont	Release și migrații	Publicare selectivă a designului
Izolare și management secrete	Escaladare către suport	Manifest operațional	Checkout și integrări validate

WooCommerce poate fi o a treia filă în același chat sau o interfață separată în WordPress/portalul HostGate. Autentificarea trebuie adaptată fiecărei intrări; drepturile unui manager de magazin nu trebuie transformate automat în acces de cPanel sau shell.

04. Arhitectura comună și granițele de încredere



Componente și responsabilități

Pluginul de chat afișează conversația, progresul, resursa curentă, preview-ul, planul și butoanele de aprobare/anulare. Nu păstrează chei principale și nu trimite o comandă root de executat. **Backendul HostGate** autentifică cererea, rezolvă contul, proiectul și serverul, aplică drepturile și înregistrează jobul.

Coadă și dispecerul gestionează concurența, ordinea pe resurse, timeout-urile și reconectarea. **Launcherul local** are un contract restrâns: primește un job autorizat și pornește numai runtime-ul aprobat, cu identitatea, mediul și limitele stabilite de platformă. Nu expune o funcție generică „execută orice shell ca orice user”.

Engine adapter pornește/reia/oprește agentul și normalizează evenimentele. Agentul păstrează bucla proprie. **Gateway-ul LLM** autorizează și măsoară apelurile către modele. **Executorul/deployerul** aplică operațiuni exacte deja autorizate și verifică starea țintei. **Collectorul de hosting** expune doar datele necesare contului când acestea sunt inaccesibile utilizatorului normal.

Acestea sunt componente logice, nu obligația de a construi opt microservicii. Backendul, coada și adaptoarele pot începe în aceeași aplicație. Separarea identităților privilegiate de procesele agentului este însă obligatorie pentru obiectivul propus.

Cele trei zone de acces

Zona HostGate conține aprobările, inventarul autoritativ, secretele centrale și politicile. Agentul nu o poate modifica. **Zona clientului/proiectului** conține codul, datele locale și istoricul agentului. **Zona live** conține aplicația publică și datele curente. Când produsul promite publicare controlată, accesul de dezvoltare la zona live trebuie blocat tehnic.

Un proces comun de backend, o coadă comună sau un gateway comun nu înseamnă automat un agent comun. Ceea ce nu vrem este un proces de agent căruia i se schimbă doar promptul în timp ce păstrează acces la toate conturile. HTTP versus STDIO este alegerea transportului; UID-ul, mount-urile, secretele și autorizarea determină accesul efectiv.

Distribuție pe mai multe servere

Backendul păstrează `server_id` pentru fiecare cont/proiect. Un serviciu local de încredere revendică joburi destinate serverului său, verifică identitatea și pornește procese temporare. Canalul intern este autentificat; nu se expune public un daemon de coding agent pe fiecare cont.

Sesiunea și workspace-ul rămân pe serverul potrivit sau sunt migrate explicit. Nu trimitem un mesaj de continuare către un worker care nu are starea nativă. La mutarea contului se blochează joburile noi, se încheie/reconciliază lucrările active, se migrează starea și se schimbă destinația în inventar.

05. Pluginul cPanel: interfață, autentificare, integrare

Ce este documentat și ce se construiește

cPanel documentează înregistrarea pluginurilor prin `install.json` și integrarea în Jupiter, inclusiv pagini proprii bazate pe LiveAPI. UAPI oferă funcții pentru datele și setările contului. Aceste mecanisme permit construirea interfeței propuse, dar nu furnizează chatul ori agentul HostGate. [S01-S05]

Meridian versus Jupiter: ghidul de pluginuri verificat descrie explicit Jupiter. Pentru interfața nouă din captura utilizatorului nu presupunem că toate extensiile și mecanismele de afișare sunt identice. Echipa trebuie să verifice metoda suportată de integrare în versiunea/tema activă; backendul și engine-urile pot rămâne independente de această diferență. Notele Meridian sunt o sursă separată, nu dovada compatibilității oricărui plugin Jupiter. [S07]

Flux de autentificare propus

Utilizatorul intră în cPanel prin mecanismul existent, inclusiv din WHMCS când este disponibil. Pagina pluginului folosește contextul autentificat pentru a obține o sesiune HostGate scurtă, legată de cont și drepturi. Backendul validează acest schimb server-side și rezolvă proprietatea proiectului din inventar.

Browserul nu stabilește autoritativ `linux_user`, `uid`, calea de lucru sau identitatea WHM. Nu se acceptă un simplu câmp ascuns cu username. Nu se transmit cookie-uri cPanel, tokenuri de sesiune sau parole către LLM. Modelul primește un identificator și contextul minim necesar.

Pentru mai multe roluri în același cont, drepturile trebuie verificate pe actorul autentificat, nu doar pe proprietarul contului. O autentificare WordPress de editor sau manager nu autorizează automat dezvoltare cu shell. Pentru sesiuni expirate, suspendări și schimbări de proprietar, aprobările viitoare sunt reevaluate.

cPanel documentează mai multe metode de autentificare și UAPI prin sesiune, LiveAPI ori CLI. Nu automatizăm clickurile pe pagini interne ca API; folosim operațiuni documentate, cu verificarea rezultatului. [S03, S04]

Interfața minimă

Antetul arată contul, proiectul, domeniul, mediul și profilul. Conversația arată separat constatări, planuri, acțiuni și rezultate. Pentru o modificare apare un rezumat explicit: resursa, ce se schimbă, efectul probabil, verificarea și posibilitatea de revenire. Utilizatorul nu trebuie să ghicească dacă agentul lucrează pe live sau pe test.

Preview-ul aplicației rulează pe un origin separat, fără cookie-urile cPanel. Nu încărcăm cod generat sau HTML neîncredere în aceeași origine privilegiată cu panoul. Ieșirea agentului și logurile sunt escapate, filtrate și limitate; nu afișăm HTML brut ori secvențe de terminal ca elemente active ale interfeței.

Chat asincron, nu request PHP lung

Pagina pluginului trimite cererea și primește `job_id`. Un worker o execută, iar interfața urmărește progresul printr-un canal autentificat de evenimente sau polling. Închiderea tabului nu trebuie să lanseze din nou lucrarea și nici să lase un proces fără proprietar.

Nu este obligatoriu ca frontendul să aibă conexiune directă la agent. SSE, WebSocket sau polling sunt opțiuni ale aplicației HostGate; autentificarea, reconectarea și limitele de buffering trebuie testate în fața proxy-urilor reale.

06. Profiluri de execuție și read-only real

Matrice de permisiuni propusă

Profil intern HostGate	Ce permite	Ce nu permite
inspect	Date filtrate, loguri aprobate, cod fără secrete, verificări limitate	Scriere live, schimbări DNS/mail, SQL de modificare
develop	Cod și teste în workspace, dependențe aprobate, DB de test	Credențiale live, root, alte conturi
store_ops	Abilities/operațiuni de magazin autorizate	Shell arbitrar pe live, administrare server

Profil intern HostGate	Ce permite	Ce nu permite
ap <code>l</code> y	Aplicarea exactă a unei operațiuni/versiuni aprobate	Interpretarea liberă a unei noi cereri de către LLM
host_ops	Interfață internă pentru echipa HostGate	Acces direct al clientului sau moștenire dintr-un chat de cont

Aceste profiluri sunt politici HostGate, nu valorile native ale opțiunii - `sandbox`. De exemplu, `inspect` poate folosi și sandboxul `read-only` al engine-ului, dar include suplimentar restricții privind DB, API-urile, rețeaua și datele citite.

De ce promptul nu este suficient

„Nu modifica nimic fără să întrebi” este o instrucțiune utilă, nu o limitare de sistem. Un agent care are acces la PHP, parolele DB și un shell poate produce efecte fără să modifice un fișier montat `read-only`. O cerere HTTP, un SQL sau încărcarea unui plugin pot avea efecte secundare.

În profilul strict `inspect`, propun instrumente limitate de colectare și credențiale fără drept de modificare, unde sunt necesare. Nu expunem generic `wp eval`, interpretoare arbitrare sau întreaga autentificare a contului. Codex documentează sandboxuri distincte, iar OpenCode avertizează că permisiunile sale nu sunt o barieră de securitate. [S18, S21, S37]

Un diagnostic nu trebuie să execute cod suspect din site ca root. Când un site este compromis, inclusiv fișierele, pluginurile, instrucțiunile proiectului și logurile se tratează ca intrări neîncredere. Pentru diagnosticarea codului, folosim copii/snapshot-uri și execuție izolată cu resurse limitate.

Nu orice citire este fără cost ori efecte

Un request către homepage poate declanșa logică de aplicație; un query de citire poate consuma multe resurse. În consecință, „`read-only`” în produs trebuie definit prin acțiunile permise și efectele acceptate, nu printr-o promisiune absolută că nimic în întregul sistem nu se schimbă. Cache-urile, acces logs și procesele de observare pot produce activitate incidentală; aceasta se documentează.

Pentru verificări HTTP active, definim domeniile și rutele permise, limita de cereri, timeout-ul și interdicția de a accesa endpointuri cu efecte de business. Pentru SQL de diagnostic, preferăm interogări prestabilite, drepturi limitate și limite de execuție.

Două niveluri de produs

Nivel pilot de încredere: agentul are accesul contului; clientul își asumă că poate modifica propriile date. Prompturi, backup și `sandbox` reduc unele riscuri, fără promisiune de aprobare imposibil de ocolit.

Nivel comercial controlat, recomandat: agentul de dezvoltare nu poate modifica live direct; operațiunile live trec prin executor autorizat. `Read-only` și aprobările sunt susținute de separarea resurselor. Nu trebuie confundate aceste două niveluri în marketing sau în criteriile de acceptare.

07. Sysadmin pentru contul cPanel

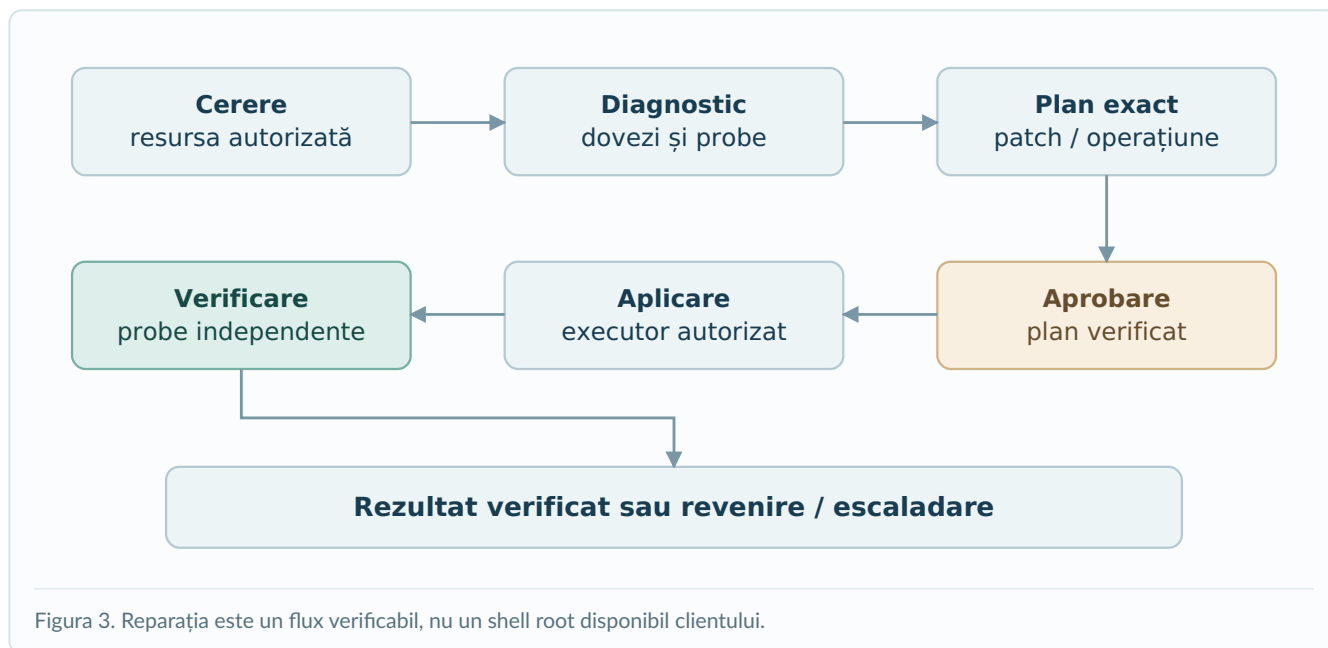
Ce poate fi produsul

Sysadminul de cont investighează și remediază probleme în limitele resurselor clientului. Nu este un administrator general al mașinii. cPanel și WHM separă funcțiile de management al site-ului de administrarea serverului; instrumentele oferite clientului trebuie să păstreze această distincție. [S03, S36]

Domeniu	Diagnostic inițial	Remediere controlată propusă
Erori web/PHP	Status HTTP, loguri proprii, configurație și versiuni accesibile	Patch sau configurare de cont, test și publicare
Spațiu și inode-uri	Inventar cu limite, directoare mari, cote	Curățare dintr-un manifest exact, nu ștergere recursivă ghicită
WordPress	Integritate, compatibilitate și erori	Actualizare/dezactivare ținută pe test și aplicare verificată
DNS și SSL	Stare publică și configurație deținută	Operațiuni suportate de cont, cu confirmarea zonei/domeniului
Email	Configurație, cote și loguri filtrate de cont	Setări permise; modificările cu risc de pierdere se confirmă separat
Aplicații proprii	Manifest, procese proprii, loguri și health checks	Release nou sau restart al aplicației proprii, dacă runtime-ul îl permite

Tabelul este un catalog de produs propus. Fiecare remediere trebuie mapată la o operațiune disponibilă și verificată pe server. Nu presupunem că toate acțiunile sunt native UAPI sau că fiecare pachet de hosting le permite.

Flux complet: eroare 500



Clientul selectează site-ul și spune „Nu mai funcționează”. Platforma autorizează proiectul și pornește inspect. Agentul citește extrase de log și configurația relevantă, apoi cere alte verificări dacă ipoteza nu este suficient susținută. Nu se presupune din start că trebuie mărită memoria, schimbată versiunea PHP sau dezactivată securitatea.

Dacă eroarea este în cod, agentul pregătește un patch pe copie. Verificatorul rulează testele și probele HTTP stabilite. Planul descrie fișierele afectate și riscul; aprobarea se leagă de hash-ul patchului și versiunea de bază. Executorul verifică faptul că fișierele live nu s-au schimbat între timp, aplică și rulează verificările post-schimbare.

Dacă problema ține de infrastructura comună — un serviciu DB indisponibil, limite globale sau o eroare a serverului — sysadminul de cont nu o „rezolvă” cerând root. Pregătește dovezi și o solicitare de intervenție pentru HostGate. Ul trebuie să spună clar că problema a fost identificată, dar remediarea necesită operatorul infrastructurii.

Remedieri fără scriere de cod

Pentru acțiuni repetitive aș prefera operațiuni deterministe: citirea cotei, listarea configurației PHP permise, verificarea unui certificat, configurarea unei valori aprobate. Agentul decide succesiunea investigației, nu reinventează apelul de administrare de fiecare dată.

Nu aș impune ca fiecare problemă rară să aibă de la început o funcție dedicată. Modul de dezvoltare poate analiza și pregăti patchuri noi în mediu izolat. Diferența este că rezultatul se verifică și se publică printr-un canal controlat, în loc să se ofere shell nelimitat pe live ca soluție universală.

Acțiuni excluse din chatul clientului

Nu oferim restart general Apache/MySQL/Exim, modificarea firewallului global, acces la /etc ori loguri complete ale tuturor clienților, schimbarea limitelor altor conturi sau instalări de pachete de sistem. Nu activăm sudo generic. O operațiune necesară la nivel de server aparține profilului intern HostGate, nu unei extensii ascunse a drepturilor clientului.

Pentru date globale utile diagnosticului, collectorul de încredere produce extrase filtrate. Nu transmite întregul log ca agentul să aleagă ulterior doar liniile clientului. Filtrarea după un nume de domeniu găsit în text nu trebuie considerată singurul mecanism de verificare a proprietății.

Prevenție, nu doar reparație

După remediare, salvăm incidentul, cauza susținută de dovezi, schimbarea aplicată, verificările și eventualul rollback. Pentru probleme recurente, operațiunea testată poate deveni un tool HostGate reutilizabil. Nu promovăm automat cod generat de agent într-un executor privilegiat.

Pentru incidente de compromitere, izolăm și păstrăm dovezile înainte de curățare. Nu pretindem „site curat” doar fiindcă s-a eliminat fișierul vizibil; verdictul trebuie să reflecte verificările efectiv efectuate și limitele lor.

08. App Builder în același plugin

Scop și experiență

Clientul descrie o aplicație: „Vreau un formular de cereri cu administrare și export”. Platforma verifică tipul aplicației, datele necesare, runtime-ul disponibil și limitele pachetului. Agentul folosește un template aprobat, dezvoltă în workspace, rulează teste, arată preview și propune publicarea.

După lansare, același chat poate răspunde la „adaugă un câmp” și la „de ce apare eroarea asta?”. Prima cerere folosește dezvoltare, a doua diagnostic. Contextul operațional este manifestul aplicației, nu presupunerea că agentul își amintește perfect toate deciziile din istoricul chatului.

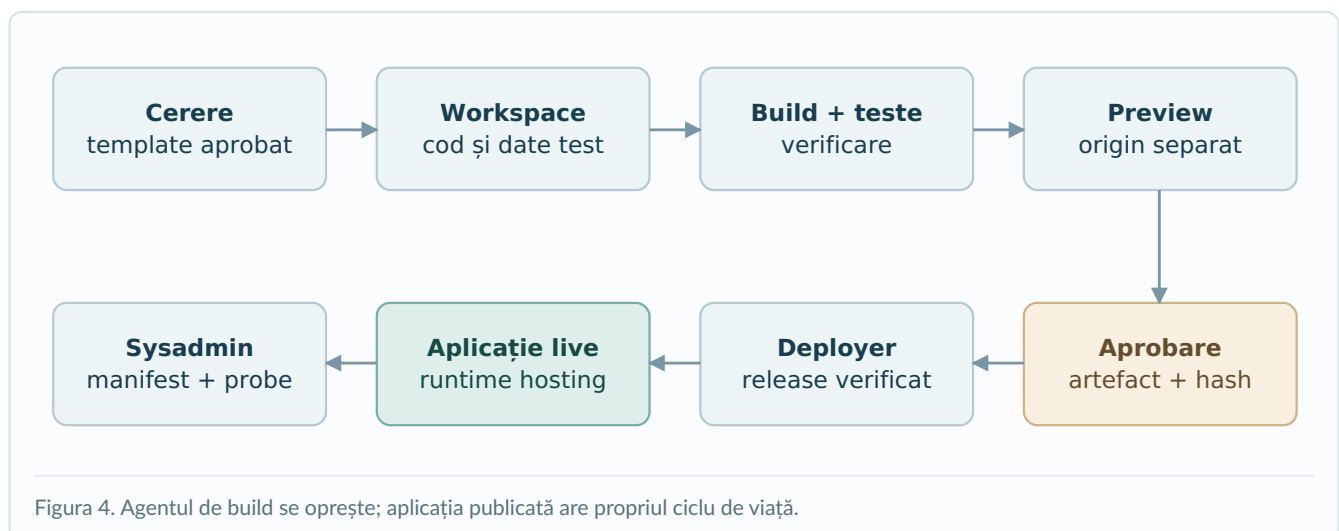
Catalog de runtime-uri

Clasă propusă	Condiție înainte de ofertare	Ce nu presupunem
Site static / frontend compilat	Build izolat și hosting static configurat	Că un dev server trebuie ținut permanent
PHP cu DB	Versiuni/extensions validate, DB și migrații gestionate	Că toate frameworkurile și pluginurile sunt compatibile
Node.js	Manager de aplicații/runtime disponibil, porturi și restart gestionate	Că orice server Node poate rula arbitrar în shared hosting
Python/Ruby	Integrare de deployment confirmată pe server	Că existența interpretorului în PATH înseamnă suport de hosting
Servicii suplimentare	Ofertă și izolare dedicate	Că se poate instala Docker, Redis sau un daemon global ca user de cPanel

cPanel Application Manager documentează deployment cu Passenger și cerințe de instalare/activare din partea providerului. Nu este o garanție că toate serverele HostGate sau toate modulele de rulare sunt deja suportate. Nu modificăm global versiunea Node pentru a satisface o singură aplicație fără evaluarea impactului. [S06]

Pentru prima versiune aş restrânge catalogul la tehnologiile pe care echipa le poate testa, monitoriza și actualiza. Un template aprobat este un punct de pornire, nu o promisiune că orice cod generat peste el este automat sigur.

Fluxul de dezvoltare și publicare



Workspace-ul conține cod și date de test, nu copia brută a tuturor datelor personale din producție.

Dependențele sunt instalate cu rețea controlată, limite de resurse și fișiere de lock. Scripturile de instalare/build se tratează ca execuție de cod neîncredere.

Verificatorul rulează comenzile de build/test aprobate pentru template, probele funcționale și verificări de bază pentru secrete. Pentru o aplicație care colectează date, include testarea autorizării, validării inputului și izolării sesiunilor; un build reușit nu demonstrează aceste proprietăți.

Preview-ul are adresă temporară, autentificare și origin separat. Nu se expune un port ales liber de agent către internet. Managerul de preview alocă ruta și resursele, apoi le șterge la expirare. Indexarea publică și livrarea reală de email/plăți sunt dezactivate în mediul de test.

Artefact și release

Rezultatul aprobat este o versiune de cod sau un artefact identificat prin hash, împreună cu manifestul, probele și migrațiile necesare. Deployerul nu primește „rulează ce spune agentul”, ci un plan cu ținte și pași verificați. Codul clientului nu se execută ca root în timpul instalării sau al testelor.

Unde stackul permite, publicarea folosește directoare de release și schimbarea controlată a versiunii active. Nu presupunem că symlink-uri sau mutări de directoare funcționează identic pentru toate combinațiile cPanel, web server și runtime; metoda este definită per clasă de aplicație.

Codul se poate inversa mai ușor decât datele. O migrație care șterge un câmp sau transformă ireversibil date necesită tratament separat. Aș prefera evoluții compatibile înainte/înapoi și validarea migrațiilor pe test. Revenirea la release-ul anterior nu implică automat revenirea bazei de date.

Agent temporar versus aplicație persistentă

Agentul de build trebuie să se oprească după job. Aplicația clientului poate rămâne pornită potrivit runtime-ului de hosting. Acestea sunt două cicluri de viață diferite. Un server Node public nu trebuie să depindă de faptul că procesul Codex sau OpenCode a rămas deschis.

Jobul temporar poate include procese de build, test și preview. Procesele destinate producției sunt preluate explicit de managerul aplicației; nu sunt lăsate accidental ca procese copil orfane. Pentru workers/cronuri aplicației, definim proprietarul, limitele, restartul și logurile.

Manifestul operațional

Pentru fiecare aplicație salvăm: runtime și versiune suportată; repository/artefact; domenii; director public; comandă de build validată; mod de pornire gestionat; DB asociată; secrete prin referință; health checks; joburi periodice; versiunea activă; metoda de rollback și schema datelor.

Acest manifest este validat și păstrat de platformă. Agentul poate propune o schimbare, dar nu poate modifica dintr-un fișier de proiect identitatea contului, limitele hostingului sau dreptul de publicare.

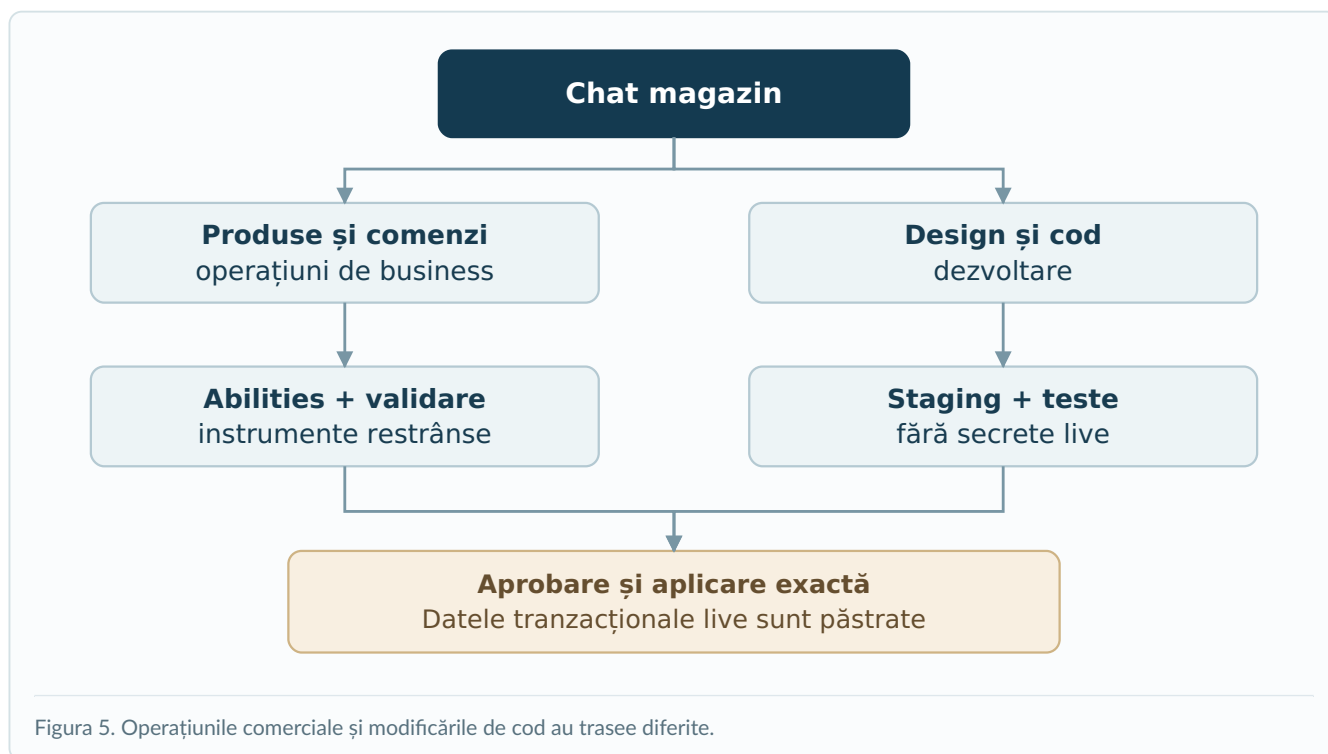
09. WooCommerce: același motor, alt domeniu operațional

De ce este diferit

WooCommerce are entități și efecte de business proprii: produse, variații, stoc, comenzi, statusuri, note, emailuri și integrări de plată/livrare. Aici nu este suficient ca un agent să știe să editeze PHP. Schimbările trebuie să respecte logica magazinului și pluginurilor.

WordPress oferă Abilities API, iar WooCommerce documentează integrarea și operațiunile aferente. Partea MCP este încă marcată developer preview în documentația consultată. Setul disponibil trebuie descoperit în instalarea și versiunea folosite, nu presupus dintr-un exemplu. [S26-S31]

Două trasee de execuție



Pentru „schimbă prețul produsului” folosim operațiunea de produs, nu un script SQL nou. Pentru „refă pagina de produs”, folosim tema/blocurile și codul din staging. Dacă o cerere le combină, planul le separă și explică ce se aplică imediat, ce necesită preview și ce trebuie confirmat.

Modulul HostGate WordPress

Aș construi un plugin cu abilities pentru contextul magazinului, branding, secțiuni, importuri asistate și publicarea unor modificări structurate. Pluginul validează inputul și permisiunile, apoi folosește logica WordPress/WooCommerce. Nu modificăm nucleul WordPress pentru a adăuga aceste operațiuni. [S27, S30]

Pentru design, recomand o temă și un set de blocuri/șabloane susținute, nu promisiunea de a opera orice builder vizual și orice temă din ecosistem. Sistemul poate genera și cod nou, dar acel traseu aparține dezvoltării izolate și testate.

Rolul WordPress este separat de UID-ul Linux. Într-o integrare locală, WP-CLI permite selectarea utilizatorului WordPress; acest parametru nu este o limită de securitate dacă agentul are shell și poate executa alte comenzi. În profilul comercial strict, agentul store_ops primește instrumente restrânse, nu acces complet la instalare. [S26, S37]

Catalog funcțional de lansare

Funcție propusă	Verificare înainte de aplicare
Creare/editare produse	Schemas, tip produs, SKU, câmpuri permise și draft/publicare

Funcție propusă	Verificare înainte de aplicare
Import asistat	Mapare coloane, rânduri invalide, duplicate, preview și idempotency
Prețuri/stoc	Produsele exacte, valorile vechi/noi și modificări concurente
Citire comenzi	Permisuni, minimizarea datelor personale și filtrare
Statusuri/note	Status valid și efectele posibile asupra emailurilor/integrărilor
Design și conținut	Staging, verificare vizuală și publicare selectivă
Plată/livrare/facturare	Integrare specifică testată; onboarding și secrete gestionate separat

Tabelul descrie capabilitățile produsului de implementat, nu afirmă că fiecare rând există deja în MCP-ul standard. Operațiuni precum rambursările, ștergerile definitive și modificarea credențialelor de plată ar trebui excluse din prima versiune sau tratate prin fluxuri cu autorizare separată.

Regula pentru publicarea magazinului

Nu suprascriem baza live cu baza din staging. WooCommerce avertizează că astfel se pot pierde comenzile noi. Publicăm doar fișierele și schimbările de conținut/configurație aprobate, iar datele tranzacționale rămân în live. [S29]

Pentru un import întrerupt, salvăm progresul operațiunii și reconciliem SKU-urile/identificatorii aplicați. Nu reluăm orbește întregul import și nu ne bazăm doar pe textul agentului „am făcut primele produse”. Executorul păstrează evidența autoritativă.

Pentru staging, oprim emailurile reale, webhookurile de producție, plățile reale și sincronizările către sisteme externe. Datele de test sunt limitate; nu oferim automat întregul istoric al clienților modelului sau workspace-ului.

10. Instrumente locale, Abilities și MCP

Separarea corectă a conceptelor

Engine-ul gestionează bucla agentică. **Modelul LLM** este serviciul de inferență apelat de engine. **Tool-ul** este o operațiune executabilă cu input și output. **WordPress Ability** este un tip de operațiune în WordPress. **MCP** este una dintre interfețele de expunere a instrumentelor. Nici co-locarea pe server, nici MCP nu oferă automat drepturi ori sandbox.

Pentru sysadmin și aplicații generale nu trebuie să instalăm WordPress numai ca să avem Abilities. Aș construi un catalog HostGate de operațiuni cu JSON Schema, autorizare, limite și audit. Pentru WordPress, adaptorul acelui catalog poate apela abilities existente sau proprii.

Catalog unificat propus

Nume propus	Implementare din spate	Profil
hosting.inspect_site	Date de cont, loguri filtrate, probe limitate	inspect
hosting.plan_repair	Plan structurat și verificabil	inspect/develop
hosting.apply_change	Executor cu aprobare validă	apply
app.prepare_workspace	Template și copie autorizată	develop
app.run_checks	Comenzi de test ale template-ului	develop
app.publish_release	Deployer și manifest validat	apply
store.inspect_catalog	WordPress/WooCommerce	store_ops
store.apply_product_plan	Ability proprie/executor	apply/store_ops

Toate numele din tabel sunt convenții de proiectare. Nu trebuie transmise către shell ca și cum ar exista deja. Implementatorul mapează fiecare operațiune la un API nativ, un helper local ori o funcție proprie testată.

Apel local prin WP-CLI

După instalarea unei versiuni compatibile a WP-CLI și într-o instalare WordPress autorizată, comenzile documentate includ:

```
wp ability list --format=json --user=hostgate_agent
wp ability get core/get-site-info --format=json --user=hostgate_agent

printf '%s' '{"fields":["name","version"]}' |
wp ability run core/get-site-info \
  --input=- --format=json --user=hostgate_agent
```

hostgate_agent este un utilizator WordPress care trebuie creat și autorizat. Numele ability și schema se descoperă în instalare. --input=- citește JSON din stdin. Exemplele arată mecanismul de integrare, nu un launcher cu izolare inclusă. [S26, S38]

Agentul poate primi instrucțiuni pentru a descoperi și folosi tool-urile înainte de a genera cod. Instrucțiunile sunt context de lucru; politicile obligatorii se impun în executor și în mediul procesului.

Când MCP merită păstrat

MCP devine util când dorim tools native cu scheme și interoperabilitate între agenți, conectare din clienți externi sau separarea executorului de filesystemul agentului. Există transport local STDIO; nu este obligatoriu HTTP public. [S28]

Prin CLI, agentul folosește instrumentul de shell pentru a invoca operațiunile. Prin MCP, acestea pot apărea ca tools distincte. Nu aş declara că una este universal mai rapidă: bootstrap-ul WordPress, numărul de apeluri, batch-urile și modelul de lucru contează. Se măsoară pe instalarea reală.

Pentru un profil fără shell general, un set de tools restrânse poate fi expus prin MCP local sau prin altă interfață suportată de engine. Nu este necesar să sacrificăm izolarea pentru a evita MCP.

11. Izolarea și execuția per utilizator

Instalare comună, procese separate

Executabilul Codex/OpenCode și workerul HostGate sunt instalate într-un director administrat de HostGate, nemodificabil de client. Launcherul pornește execuția în contul autorizat, cu UID/GID, grupuri, HOME, mediu și directoare controlate. În CageFS trebuie făcute vizibile doar runtime-urile și resursele necesare.

Codex folosește implicit `~/ .codex` pentru starea locală. Un `CODEX_HOME` configurat trebuie separat, nu comun tuturor conturilor. Pentru OpenCode se folosesc directoarele sale native de configurare/date, verificate pe release-ul ales. Stocarea engine-ului nu trebuie să fie direct în document root. [S17, S39]

CloudLinux oferă mecanisme distincte pentru CageFS și controlul resurselor. `cd /home/client` nu schimbă identitatea, iar o simplă schimbare de UID nu dovedește intrarea corectă în CageFS și LVE. Launcherul se validează cu probe de acces și de resurse pe serverul real. [S08]

Ce trebuie separat suplimentar

Mai multe aplicații sub același UID nu sunt izolate între ele doar prin directoare. Dacă produsul promite acces exclusiv la proiectul selectat, aplicăm restricții suplimentare de filesystem, procese, DB și rețea. Relația cont/proiect/profil trebuie să fie parte din politica jobului.

Garanțiile se referă la execuțiile lansate de platforma HostGate. Proprietarul contului poate avea în continuare acces prin cPanel, FTP sau SSH și poate modifica separat propriul site. De aceea, lock-urile interne nu înlocuiesc verificarea schimbărilor externe înainte de aplicare.

Separarea live/staging este aceeași problemă. Nu este suficient ca agentul să înceapă într-un alt director dacă poate deschide oricând `public_html` sau citi parolele live. Variantele sunt identități distincte, namespace/sandbox suplimentar ori workspace-uri pe workers separați. Alegerea se face după un prototip compatibil cu CageFS, nu printr-o comandă universală pretins sigură.

Lansatorul privilegiat

Un helper privilegiat poate fi necesar pentru pregătirea izolării. El acceptă doar un identificator de job și operațiuni limitate; rezolvă server-side contul și resursele. Nu încarcă biblioteci, scripturi de build sau cod PHP din contul clientului înainte de reducerea privilegiilor.

Validarea căilor trebuie să prevină traversarea, symlink-urile și schimbarea țintei între verificare și acces. Nu este suficient un test de prefix textual. Fișierele de job, binarele și politicile folosite de launcher sunt în zone de încredere, nu controlate de client.

Nu montați socketul Docker al hostului în mediul agentului pentru comoditate. Documentația Docker explică sensibilitatea accesului la daemon și necesitatea de a limita cine îl controlează. Worker-ele containerizate sunt o alternativă de izolare numai când controlul hostului rămâne în afara agentului. [S09]

Rețea, secrete, procese

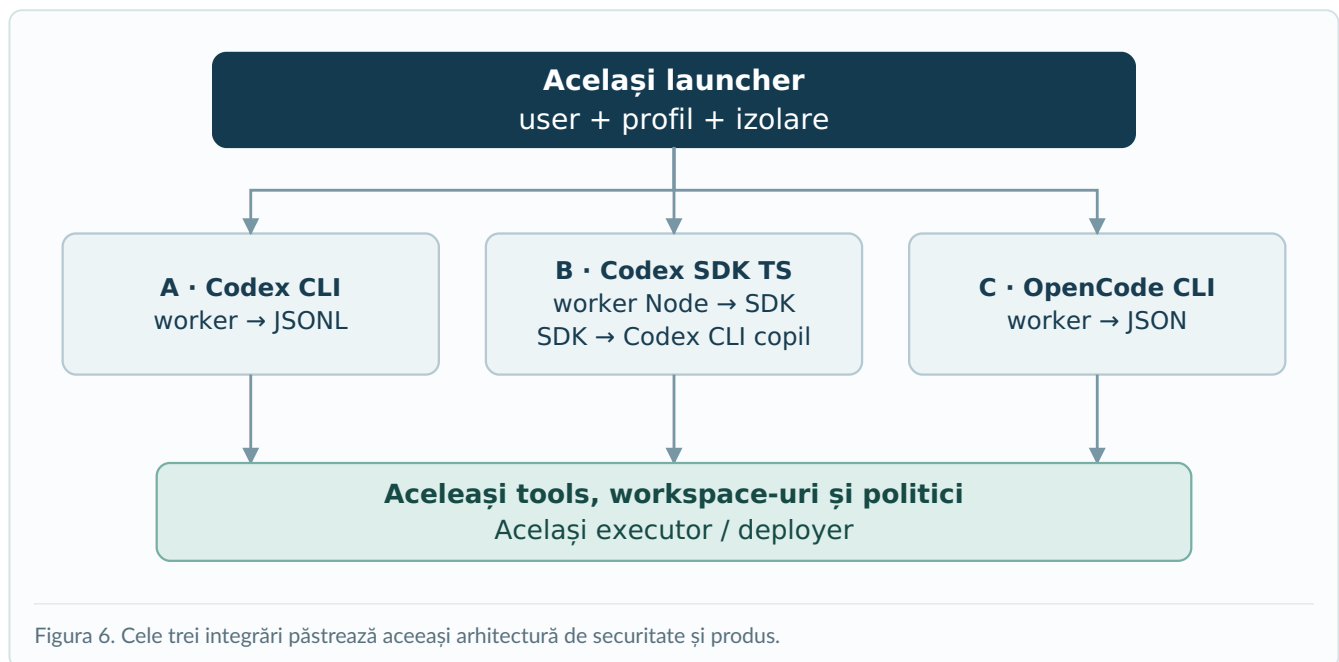
Accesul de rețea se definește per profil: gateway LLM, registries aprobate pentru build, servicii de test și probe asupra domeniilor autorizate. Nu lăsăm acces general la panouri de management, servicii interne, metadata endpoints sau alte DB doar pentru că sunt „pe același server”.

Se filtrează variabilele de mediu; nu se moștenește întregul mediu root/backend. Tokenurile de job sunt scurte, limitate și revocabile. Un secret disponibil procesului clientului nu este considerat ascuns de acel client sau de codul executat în același mediu.

La anulare, urmărim întregul arbore/cgroup al jobului. În prima versiune oprim jobul și apoi reconciliem efectele. Un proces terminat nu dovedește că o operațiune externă a fost anulată, iar un timeout nu autorizează automat retry.

12. Varianta A – Codex CLI direct

Arhitectură



Codex CLI oferă execuție neinteractivă și evenimente JSONL. Este potrivit pentru o integrare care poate porni și controla procese din PHP, Python, Node sau alt limbaj. Nu parsăm interfața interactivă ori textul colorat. [S10]

Utilizare în cele trei produse

Pentru sysadmin, Codex pornește într-un workspace de diagnostic cu tools limitate. Pentru builder, pornește în proiectul izolat cu teste și dependențe controlate. Pentru WooCommerce, primește catalogul de operațiuni și accesul potrivit profilului. Diferența dintre produse este în politici și context, nu în necesitatea de a modifica engine-ul.

Un exemplu pentru un workspace de test, după ce identitatea și izolarea sunt deja aplicate:

```
cd -- "$WORKSPACE" || exit 1

codex exec --json --sandbox workspace-write \
  -c 'approval_policy="never"' - < "$PROMPT_FILE"

# Un job ulterior: sesiunea este rezolvata de backend, nu de browser.
codex exec --json --sandbox workspace-write \
  -c 'approval_policy="never"' \
  resume "$THREAD_ID" - < "$NEXT_PROMPT_FILE"
```

Promptul complet poate veni din stdin, iar reluarea explicită folosește ID-ul sesiunii. Configurația de sandbox și aprobare trebuie fixată pentru profil. Exemplele nu sunt instrucțiuni de a porni direct pe live și nu includ launcherul CageFS. [S10, S13, S16]

Politica never evită aprobările interactive; nu înseamnă permisiune nelimitată. Pentru aprobări de business, propun separarea planificării de aplicare. Protecția bazei de date și a API-urilor nu rezultă doar din workspace-write.

Adaptorul HostGate

Adaptorul folosește un vector de argumente și pipe-uri, nu o comandă shell concatenată cu inputul utilizatorului. Citește stdout/stderr în paralel, validează evenimentele, păstrează identificatorul sesiunii și transmite doar datele relevante pentru UI. Nu presupune streaming text token-cu-token doar pentru că primește JSONL.

La final verifică exit code, evenimentele de eroare și verificatorul independent. Un mesaj „gata” nu înseamnă succes verificat. Evenimentele despre tool-uri nu autorizează o operațiune privilegiată; sunt observații din procesul agentului.

Un request întrerupt de browser nu produce un nou job. Sesiunile sunt reluate explicit, sub contul/profilul corect. Nu se folosește `--last` într-un backend multi-conversație.

Avantaje, limite și criteriul de alegere

Avantajul este o integrare directă și depanabilă, fără o bibliotecă Node obligatorie. HostGate întreține parserul și gestionarea procesului. Pentru un backend PHP, aceasta este prima variantă pe care aș prototipa-o.

Limita este interacțiunea preponderent pe joburi. Aprobările bidirecționale în timpul execuției și alte funcții bogate de UI trebuie evaluate separat. Codex App Server este o interfață alternativă pentru astfel de clienți; documentația și maturitatea traseului concret trebuie verificate înainte de producție. [S19]

Testele specifice sunt reluarea sesiunii după închiderea procesului, JSON incomplet, stderr excesiv, timeout, sandbox incompatibil cu CageFS și anularea tuturor proceselor copil.

13. Varianta B — Codex SDK TypeScript

Ce diferă față de CLI

@openai/codex-sdk este o bibliotecă server-side care controlează Codex CLI. Implementarea TypeScript consultată pornește procesul și interpretează JSONL; nu este SDK-ul general pentru a trimite prompturi către un model și nu înlocuiește agentul cu unul nou. [S11–S13]

Workerul Node trebuie lansat deja în contul/mediul corect. În sursa verificată, apelul spawn al SDK-ului nu setează un UID separat. `workingDirectory` nu transformă un worker root într-un proces al clientului. [S13]

Nucleu de integrare verificat

Următorul fragment arată apelurile SDK și nu reprezintă o aplicație completă sau un launcher securizat. Variabilele sunt rezolvate de codul de încredere al HostGate, iar erorile sunt transmise job managerului:

```
import { Codex, type ThreadOptions } from "@openai/codex-sdk";

const codex = new Codex({
  codexPathOverride: trustedCodexBinary,
  env: allowedEnvironment,
});

const options: ThreadOptions = {
  workingDirectory: authorizedWorkspace,
  sandboxMode: "workspace-write",
  approvalPolicy: "never",
};

const thread = savedThreadId
  ? codex.resumeThread(savedThreadId, options)
  : codex.startThread(options);

const streamed = await thread.runStreamed(prompt, {
  signal: abortSignal,
});

for await (const event of streamed.events) {
  // Doar catre backendul intern; filtrare inainte de browser.
  process.stdout.write(JSON.stringify(event) + "\n");
}
```

Metodele `runStreamed`, `resumeThread`, opțiunile și transmiterea `AbortSignal` sunt susținute de codul oficial consultat. Exporturile și versiunea pachetului trebuie fixate și compilate în prototip. `allowedEnvironment` înlocuiește moștenirea implicită a mediului, atunci când este furnizat. [S12, S14–S16]

Operare

Workerul și dependențele pot fi instalate comun, read-only pentru clienți. Nu este necesar un `node_modules` separat pentru fiecare cont doar pentru worker. Datele Codex și workspace-urile sunt separate. Verificăm accesibilitatea Node și a executabilului din CageFS.

HostGate continuă să gestioneze coada, identitatea, limitele și publicarea. SDK-ul reduce codul de integrare cu agentul; nu furnizează automat un control de aprobare imposibil de ocolit sau contabilitate autoritativă.

Anularea prin `AbortSignal` este o funcție utilă, dar nu înlocuiește verificarea proceselor copil și a efectelor externe. Limita maximă a jobului și oprirea întregului grup de procese aparțin infrastructurii HostGate.

TypeScript versus Python

Documentația Codex descrie și un SDK Python care controlează App Server prin JSON-RPC. Nu generalizăm arhitectura wrapperului TypeScript la orice produs numit „Codex SDK”. Propunerea B din acest document este explicit TypeScript. [S11]

Avantaje, limite și criteriul de alegere

Aș selecta varianta B când workerul este deja Node/TypeScript, iar echipa vrea tipuri și metode oficiale pentru sesiuni și streaming. Cu același runtime, model și aceleași instrumente, nu presupunem o calitate agentică superioară CLI-ului.

Trade-off-ul este dependența de pachetul SDK și runtime-ul Node. Testul corect între A și B măsoară robustețea integrării, anularea și erorile, nu pretinde că sunt două modele de raționament concurente.

14. Varianta C – OpenCode CLI per utilizator

Fără agent comun pentru toate conturile

OpenCode documentează execuția neinteractivă, ieșirea JSON și sesiuni explicite. Putem păstra aceeași fundație de joburi la cerere și izolare per utilizator. Nu este necesar un OpenCode comun care primește cereri pentru toate site-urile și are acces transversal. [S20]

Exemple de sintaxă documentată, pentru testarea adaptorului:

```
opencode run --format json "Analizeaza proiectul de test"

opencode run --format json --session "$SESSION_ID" \
  "Continua planul de modificare"
```

În backend, argumentele se transmit separat și inputul este tratat ca date. Pentru prompturi mari sau sensibile se validează mecanismul stdin al release-ului ales, evitând expunerea lor în lista proceselor.

Versiunea CLI și transportul intern

Documentația standard și cea v2 nu au toate opțiunile identice. V2 documentează inclusiv un mod - - standalone cu server privat. Nu combinăm flag-uri din generații diferite și nu presupunem că „CLI” garantează absența oricărui listener intern. [S20, S25]

Cerința de produs este ca orice componentă internă să fie izolată, temporară, accesibilă numai execuției autorizate și eliminată la final. Un listener pe localhost nu este automat privat față de ceilalți utilizatori locali. Testăm bind, autentificare, acces inter-user și cleanup.

Alegerea modelelor

OpenCode documentează integrarea cu mai mulți furnizori, inclusiv Moonshot. Acesta este un motiv real de evaluare pentru preferința utilizatorului de a putea folosi modele Kimi. Nu afirmăm compatibilitatea oricărui nume de model doar pentru că providerul este disponibil. [S23]

Codex documentează și configurări de provideri/endpointuri alternative. Aș compara compatibilitatea efectivă cu tool-urile, formatele și gateway-ul, nu aș reduce comparația la etichetele „deschis” versus „închis”. [S17]

Permisuni și configurații

OpenCode spune explicit că nu oferă sandbox de securitate. Regulile allow/ask/deny sunt utile, dar nu reprezintă izolare între conturi sau între staging și live. Aceasta trebuie impusă extern. [S21, S24]

Configurația globală și cea din proiect pot schimba comportamentul. Pentru serviciul gestionat, HostGate trebuie să controleze configurația efectivă, furnizorii, pluginurile, partajarea și actualizările. Nu încercăm automat extensii sau scripturi arbitrare dintr-un proiect ca parte privilegiată a platformei. [S22]

Aprobările în modul neinteractiv trebuie testate pe versiunea selectată. Dacă o cerere de permisiune blochează sau e refuzată, nu rezolvăm problema prin auto-aprobare globală. Pentru funcțiile comerciale sensibile folosim același model plan/aprobare/aplicare din backend.

Avantaje, limite și criteriul de alegere

Aș păstra OpenCode ca opțiune reală pentru toate cele trei produse, mai ales când alegerea providerului este importantă. Costul și rata de reușită se măsoară pe sarcini identice, cu aceleași instrumente și politici.

Costul suplimentar de integrare este validarea atentă a versiunii, a evenimentelor și a comportamentului noninteractiv. Lipsa sandboxului propriu nu îl elimină automat, dar crește importanța izolării HostGate. Nu există în acest studiu un benchmark care să dovedească un câștig universal al OpenCode sau Codex.

15. Comparație și contract de engine

Comparația arhitecturală

Criteriu	Codex CLI	Codex SDK TypeScript	OpenCode CLI
Engine efectiv	Codex	Același Codex	OpenCode
Interfață	Proces + JSONL	Bibliotecă peste CLI	Proces + JSON
Backend PHP	Integrare directă	Worker Node suplimentar	Integrare directă
Per user, la cerere	Da, prin launcher	Da, workerul sub user	Da, după validarea modului CLI
HTTP comun obligatoriu	Nu	Nu	Nu
Tools locale/Abilities	Prin integrarea configurată	Aceeași bază CLI	Prin integrarea configurată
Sandbox al produsului	Documentat; de verificat local	Cel al runtime-ului Codex	Nu este oferit ca barieră de securitate
Aprobări de business	HostGate	HostGate	HostGate
Motiv principal de alegere	Simplitate multi-limbaj	Integrare TypeScript	Flexibilitate furnizori și rezultate măsurate

Capabilitățile native din tabel sunt susținute de documentația engine-urilor. Potrivirea cu backendul și alegerea propusă sunt evaluări de proiectare. [S10–S13, S18, S20, S21]

Nu construim nouă soluții

Profil	Codex CLI	Codex SDK TS	OpenCode
Sysadmin de cont	Același catalog de diagnostic și reparații	Același catalog	Același catalog
App Builder	Același workspace și deployer	Același workspace/deployer	Același workspace/deployer
WooCommerce	Același plugin și Abilities	Același plugin	Același plugin

Diferențele engine-urilor se izolează într-un adaptor mic. Nu introducem un agent superior care să decidă fiecare tool pentru alt agent. HostGate stabilește profilul autorizat și engine-ul execută bucla în interiorul lui.

Contract intern propus

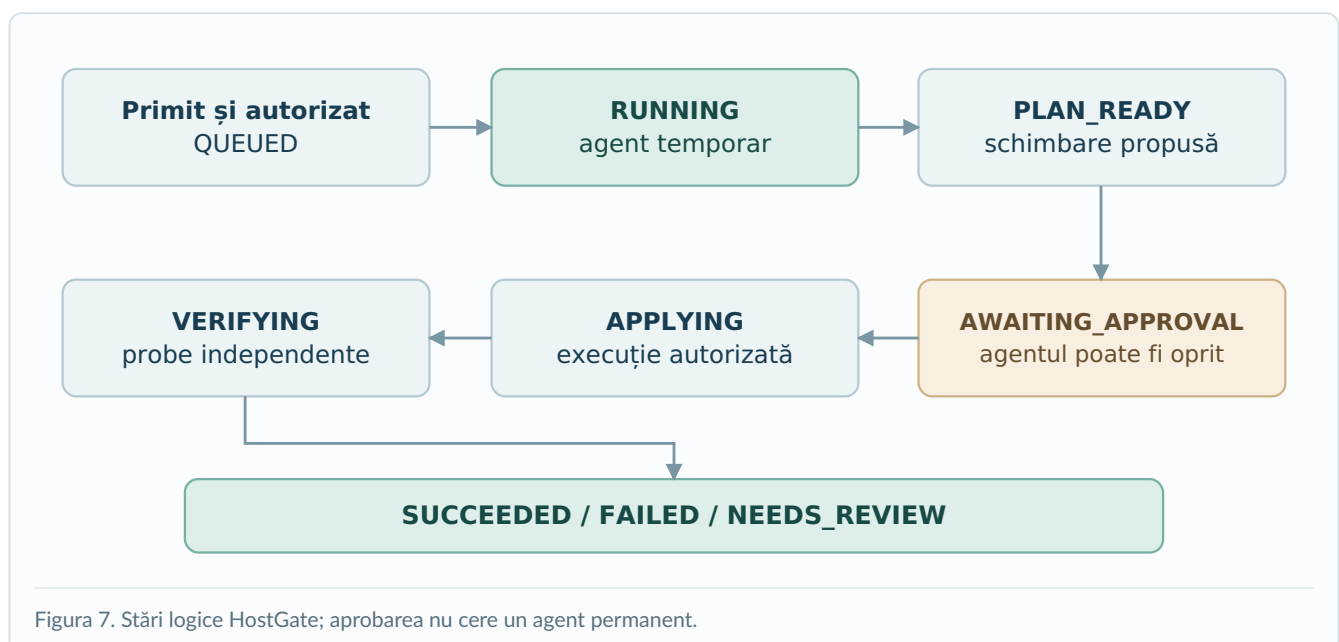
Adaptorul are responsabilitățile `start`, `resume`, `stream`, `cancel` și `finalize`. Acestea sunt nume de interfață HostGate, nu metode native identice în toate produsele. Returnează identificatorul sesiunii, mesaje pentru utilizator, rezumate ale operațiunilor, erori, consum observat și artefacte.

Evenimentele normalizate pot fi `job.started`, `message.delta`, `tool.started`, `tool.finished`, `artifact.proposed`, `job.failed` și `job.finished`. Dacă engine-ul nu oferă delte de text, adaptorul nu le inventează. Un eveniment de engine nu poate deveni `approval.granted`; aprobarea aparține exclusiv backendului.

Sesiunile native nu sunt presupuse portabile între Codex și OpenCode. La schimbarea engine-ului se creează o sesiune nouă cu rezumat, manifest și artefacte. Chiar în același engine, schimbarea profilului poate necesita sesiune nouă pentru a evita contaminarea contextului și a drepturilor.

16. Joburi, aprobări și operațiuni concurente

Ciclul unei lucrări



Aceste stări sunt propuse pentru HostGate. Un job poate fi doar de diagnostic și se încheie după verificarea raportului. O operațiune cu efecte poate trece prin aprobarea planului. Închiderea procesului agentului și închiderea operațiunii de business sunt evenimente distincte.

Aprobarea în două faze

Agentul pregătește un plan sau un artefact. Backendul îl verifică și îl prezintă. Procesul poate fi închis până la răspunsul utilizatorului. La aprobare, backendul înregistrează actorul, resursa, hash-ul planului, versiunea de bază, expirarea și limitele.

Executorul verifică din nou toate aceste condiții înainte de aplicare. Dacă live s-a schimbat, planul se reface sau se revalidează. Nu folosim un simplu text „aprobat” din conversația agentului. Dacă un plan presupune trei operațiuni, aprobarea nu acoperă automat o a patra descoperită în timpul execuției.

Pentru acțiuni cu impact extern — email, rambursare, DNS, ștergere — planul arată explicit efectul și reversibilitatea. Prima versiune poate exclude aceste acțiuni în loc să pretindă un rollback universal.

Serializare între produse

Un lock doar pe conversație nu este suficient. Sysadminul poate încerca să repare un fișier în timp ce App Builderul publică aceeași aplicație. Blocarea trebuie să urmărească resursa: cont, proiect, release, grup de produse sau operațiune de magazin.

Pentru MVP, propun o singură mutație de cod/configurație per proiect la un moment dat. Operațiunile de business pot avea chei mai fine când sunt proiectate și testate astfel. Joburile cu citire pot rula separat numai dacă nu perturbă resursele ori observațiile necesare unei verificări.

Lock-ul are lease și identificador de generație/fencing. Un worker vechi, care revine după timeout, nu mai poate publica folosind o autorizare depășită. Acest mecanism trebuie verificat în executor, nu doar în dispecer.

Retry și idempotency

Browserul trimite o cheie de idempotency pentru cererea sa. Backendul nu creează două joburi pentru aceeași retrimiteră. Executorul păstrează separat identicatorii operațiunilor aplicate. „Exactly once” nu se presupune între API-uri și procese care pot cădea; folosim deduplicare și reconciliere.

Dacă s-a schimbat un preț, s-a creat un produs sau s-a trimis un email înainte de întrerupere, retry-ul trebuie să observe efectul existent. O comandă de shell terminată cu eroare poate să fi produs deja schimbări parțiale. În această situație jobul devine NEEDS_REVIEW sau intră în reconciliere, nu se resetează automat la zero.

Stop, rollback și compensare

Stop oprește execuțiile viitoare pe cât posibil, dar nu anulează efectele trecute. Rollback poate însemna revenirea unui release sau restaurarea unui fișier. Compensarea poate însemna o operațiune de business inversă, care nu este întotdeauna posibilă sau echivalentă. Ul trebuie să distingă aceste stări.

17. Publicare și verificare independentă

Executorul nu este un shell root pentru agent

Deployerul/executorul primește operațiuni din catalog, targeturi autorizate și artefacte validate. Nu execută ca root scripturi generate, hook-uri de proiect sau PHP din magazin. Când o operațiune necesită privilegii de hosting, partea privilegiată rămâne minimă și deterministă.

Tip de modificare	Unitatea de aprobare	Verificare	Revenire posibilă
Patch sysadmin	Diff + hash fișier bază	Sintaxă, probe relevante	Copia fișierului, dacă starea permite
Release aplicație	Artefact + manifest	Build, health, teste funcționale	Release anterior; DB tratată separat
Configurație cont	Valori exacte și resursă	API read-back și probă	Valorile vechi, dacă operațiunea permite
Produce Woo	ID/SKU și valori vechi/noi	Citire rezultat și reconciliere	Schimbare inversă controlată
Comenzi/integrări	Operațiune și efecte explicite	Stare și evenimente asociate	Nu presupunem reversibilitate

Cine verifică

Verificatorul HostGate rulează probe definite pentru operațiune. Agentul poate genera teste suplimentare, dar nu poate elimina verificările obligatorii și nici valida singur succesul unei publicări sensibile. Un test ales exclusiv de autorul schimbării poate omite tocmai defectul introdus.

Pentru sysadmin, cerem dovezi înainte/după și un rezultat măsurabil. Pentru aplicații, verificăm pornirea, rutele, autorizarea și funcția cerută. Pentru WooCommerce, verificăm entitățile și păstrarea comenzilor/stocurilor concurente. Rezultatul transmis clientului separă „executat”, „verificat” și „neconfirmat”.

Backup și consistență

Backup-ul fișierelor nu este backup al bazei de date, iar un snapshot DB nu este automat sigur de restaurat peste comenzi noi. Strategia de recuperare este specifică resursei și include test de restaurare. Referința backup-ului este păstrată de HostGate, nu doar scrisă într-un mesaj de agent.

Pentru operațiuni care traversează fișiere, DB și sisteme externe, nu presupunem o tranzacție atomică globală. Planul precizează punctele de oprire, reconcilierea și efectele imposibil de inversat. Acesta este motivul pentru care importurile, migrațiile și publicarea sunt componente de produs, nu simple comenzi „apply”.

Live fără downtime promis automat

Nu promitem publicare fără întrerupere pentru orice stack. Pe template-urile suportate definim mecanismul și testăm durata/efectele. Dacă o operațiune cere mentenanță, aceasta apare în plan și în confirmare. Un sysadmin de cont nu trebuie să mascheze indisponibilitatea prin ștergerea logurilor sau dezactivarea verificărilor.

18. Date, API intern și componente de construit

Model minim de date

Entitate HostGate	Câmpuri și sens
Account	Identitate comercială, cont cPanel, server, stare și drepturi
Project	Tip, domenii, live/workspace, runtime, DB și proprietar
Conversation	Actor, proiect, profil și sesiuni native asociate
Job	Engine/model/versiuni, stare, buget, timeout, lease, idempotency
ChangeSet	Plan, ținte, hash, versiune bază, probe și efecte
Approval	Actor, scope exact, expirare, nonce și referință ChangeSet
Operation	Apel autorizat, rezultat, stare parțială și cheie de deduplicare
Artifact/Release	Hash, proveniență, manifest, teste și versiune activă
Usage/Audit	Consum din gateway, accesuri, operațiuni și retenție

Datele autoritative pentru aprobări și facturare nu se păstrează numai în home-ul clientului. Istoricul agentului poate fi modificabil de acel cont și nu trebuie tratat ca jurnal antifraudă. Putem salva separat evenimentele utile pentru investigație, cu minimizarea datelor sensibile.

API intern propus

Acesta este un contract de construit, nu un API existent al produselor:

```

POST /v1/projects/{project_id}/conversations
POST /v1/conversations/{conversation_id}/messages
GET /v1/jobs/{job_id}
GET /v1/jobs/{job_id}/events
POST /v1/jobs/{job_id}/cancel
GET /v1/changes/{change_id}
POST /v1/changes/{change_id}/approve
POST /v1/changes/{change_id}/reject

```

Toate operațiunile verifică actorul și proprietatea. Identificatorii sunt opaci, dar opacitatea nu înlocuiește autorizarea. Browserul nu primește endpointul launcherului, o funcție de alegere a UID-ului sau un terminal root.

Exemplu de job rezolvat server-side

```
{
  "schema_version": 1,
  "job_id": "job-demo-001",
  "account_id": "account-demo",
  "project_id": "project-demo",
  "server_id": "hosting-demo",
  "profile": "develop",
  "engine": "codex-cli",
  "engine_version_ref": "approved-engine-manifest",
  "workspace_ref": "workspace-demo",
  "session_ref": "authorized-session-reference",
  "policy_ref": "develop-v1",
  "budget_ref": "approved-job-budget",
  "request_id": "idempotency-demo"
}
```

Exemplul este intenționat fără parole și comenzi shell. Referințele sunt rezolvate în inventar; nu acceptăm o cale absolută arbitrară doar fiindcă agentul a inclus-o în JSON.

Pachet de implementare

Echipa construiește pluginul/UI și integrarea de autentificare; job managerul; launcherul validat; un adaptor de engine inițial; catalogul de tools de cont; workspace/deployer pentru aplicații; pluginul WordPress și tema susținută; gateway-ul de modele; verificatorul și auditul.

Se refolosesc bucla agentică și sesiunile native. Nu este nevoie de un orchestrator cu mai multe LLM-uri doar pentru a trimite o cerere către agent. O interfață deterministă între componente este suficientă pentru primul produs.

19. Modele, chei, consum și capacitate

Cheia principală rămâne în afara contului

Propun un gateway HostGate care primește tokenuri scurte legate de job, validează providerul/modelul și bugetul, apoi adaugă credențialele reale. Gateway-ul nu este o a doua buclă agentică; autorizează și transportă apelurile.

Codex documentează autentificare pentru automatizări și configurarea endpointurilor. Expunerea cheii principale în mediul în care rulează cod de proiect nu trebuie considerată sigură. Configurația exactă a gateway-ului se testează cu protocolul, streaming-ul și tool calling-ul necesare engine-ului. [S10, S17, S35]

Tokenul de job trebuie să aibă limită, expirare și posibilitate de revocare. Îl tratăm ca accesibil mediului clientului. După finalizare nu poate fi folosit ca abonament nelimitat către furnizor. Limitele nu se bazează exclusiv pe un câmp de consum declarat de agent.

Politica de modele

Selecția modelului se configurează în catalogul aprobat și poate diferi între sarcini. Nu promitem că un model mai ieftin per token costă mai puțin pe rezultat. Măsurăm încercările, durata, erorile și suportul necesar.

Nu presupunem că sesiunile își păstrează identic comportamentul după schimbarea modelului sau runtime-ului. Salvăm versiunea configurației pentru fiecare job și facem rollout controlat. Nu actualizăm simultan toate engine-urile, tool-urile și pluginurile în producție.

Capacitate

Dimensionarea se face după joburi active și procesele lor copil, nu după numărul total de conturi. Măsurăm memoria întregului job, CPU, I/O, durata WP-CLI, numărul de procese și impactul asupra site-urilor găzduite. Buildurile și testele de browser pot fi mai costisitoare decât un diagnostic simplu.

Sunt necesare limite pe cont, proiect, server și provider. Coadă oferă echitate, astfel încât un client cu multe mesaje să nu ocupe toate sloturile. Se pot rezerva clase separate de resurse pentru diagnostic scurt și builduri grele; nu atribuim valori înaintea pilotului.

Un mod de diagnostic determinist de bază poate funcționa și când providerul LLM este indisponibil. În acest caz UI afișează datele vericatorului, nu inventează o concluzie AI. Joburile cu efecte nu sunt reluate automat după o eroare de provider fără verificarea stării.

Costuri de urmărit

Costul total per sarcină verificată include inferența, timpul de worker/build, stocarea/preview, operațiunile externe și minutele de suport. Raportăm separat consumul nereușit. Pragurile de buget și condițiile abonamentului se aleg după măsurători, nu după o estimare prezentată ca fapt.

20. Securitate, confidențialitate și incidente

Amenințări principale și controale

Risc	Control cerut
Cerere falsificată pentru alt cont	Rezolvare server-side și verificare la fiecare operațiune
Prompt injection din loguri/cod/produse	Date neîncredere, tools limitate, nicio extindere de drepturi din conținut
Acces live din staging	Izolare filesystem, DB, rețea și secrete; test negativ
Ocolirea read-only prin PHP/API	Profil fără interpretoare arbitrare și fără credențiale de scriere
Exfiltrare de secrete	Minimizarea contextului, filtrare, egress limitat și gateway
Configurație de agent modificată de proiect	Configurație gestionată; extensii și hooks controlate
Operațiuni duplicate după retry	Idempotency, jurnal de operațiuni și reconciliere
Dependențe/build malițios	Execuție izolată fără privilegii, lockfiles și registries aprobate
Preview cu acces la cPanel	Origin separat și fără cookie-uri/tokenuri de management
Artefact schimbat după aprobare	Hash și versiune bază revalidate la aplicare

Tabelul reprezintă amenințări și controale de proiectare, nu un audit final de securitate. Testele adversariale trebuie efectuate pe implementarea reală. Nici sandboxul engine-ului, nici CageFS nu se presupun suficiente pentru toate aceste riscuri.

Minimizarea datelor

Pentru diagnostic trimitem extrase relevante de log, nu întregul home. Pentru WooCommerce trimitem câmpurile necesare operațiunii, nu automat adrese, telefon și istoricul complet. Pentru dezvoltare folosim date sintetice sau copii anonimizate, unde este posibil.

Secretele sunt referințe în manifest, nu valori în conversație. Rapoartele afișează ce surse au fost citite și ce verificări s-au făcut, fără să includă parole. Politicile de retenție și acces la conversații trebuie stabilite separat pentru client, suport și administratori.

Înainte de lansare se verifică termenii providerilor, utilizarea comercială permisă, acordurile privind datele și condițiile componentelor distribuite. Documentul nu este analiză juridică și nu presupune că un abonament personal de chatbot poate fi împărțit tuturor clienților.

Răspuns la incident

HostGate trebuie să poată dezactiva un engine, un model, un tool, publicarea sau un cont fără a opri întregul hosting. Revocă tokenurile de job, oprește procesele relevante, păstrează auditul și reconciliază operațiunile.

La un incident de acces între conturi, rollout-ul este oprit. La o regresie de engine, se revine la versiunea validată sau se suspendă funcția afectată. Nu se actualizează politicile „din mers” doar în prompt pentru a masca un defect de izolare.

Un agent de sysadmin nu este un substitut pentru monitorizare, backup, management de vulnerabilități și echipa de intervenție. Produsul poate automatiza o parte din investigație și execuție, dar responsabilitatea operațională rămâne în platformă și la provider.

21. Teste de acceptare și benchmark obiectiv

Teste funcționale pe fiecare produs

Sysadmin: eroare PHP cu dovadă clară; problemă de configurație a aplicației; cotă atinsă; resursă inaccesibilă din cont; incident care cere infrastructura comună. Pentru fiecare, definim cauza reală, schimbările permise și criteriul de succes înainte de rulare.

App Builder: aplicație din template, modificare de schemă, login cu roluri, build eșuat, dependență incompatibilă și publicare cu rollback de cod. Evaluăm funcția cerută, nu doar aspectul preview-ului.

WooCommerce: produs simplu/variabil, import cu duplicate, editare preț/stoc, status de comandă, schimbare de homepage și publicare în timpul apariției unei comenzi noi. Se verifică efectele pluginurilor susținute și interdicțiile din staging.

Matrice de securitate și fiabilitate

Test	Rezultat obligatoriu
Agentul A încearcă să citească B	Refuz de sistem, nu doar refuz în limbaj natural
Browserul falsifică project/session ID	Refuz înainte de pornirea jobului
Editor WP cere shell cPanel	Nu primește drepturi suplimentare
Două aplicații au același UID	Limitele promise per proiect rămân efective
Agentul inspect încearcă DB write	Operațiunea este blocată
Sysadmin și builder modifică aceeași aplicație	Serializare și revalidare
Procesul moare în timpul apply	Stare parțială identificată; fără retry orb
Tabul se închide și se redeschide	Aceeași lucrare, fără dublare
Worker vechi revine după lease expirat	Nu mai poate publica
Plan modificat după aprobare	Aprobarea este invalidată
Intră o comandă Woo în timpul deploy-ului	Comanda este păstrată
Build încearcă acces la chei/live	Acces blocat
OpenCode deschide listener intern	Inaccesibil altor clienți și închis la final
Stop în timpul buildului	Procesele temporare sunt oprite și efectele raportate
Providerul refuză bugetul	Terminare controlată, fără retry nelimitat
Agentul afirmă succes, testul eșuează	Jobul nu este „succes verificat”

Metoda de comparație

A și B se testează cu același runtime Codex, model, tools și stare inițială. Asta compară integrarea. OpenCode se testează cu același model când compatibilitatea permite, apoi cu modelele candidate proprii; rezultatele nu se confundă între engine și model.

Fiecare caz se repetă suficient pentru a observa variația, cu resetarea stării de test. Echipa fixează numărul de rulări și pragurile înaintea selecției. Nu este stabilit aici un procent de reușită imaginar sau o capacitate de producție neobservată.

Metricile sunt: rezultat verificat, schimbări nedorite, cost total, timp, încercări, intervenție manuală și corectitudinea raportului. Orice încălcare a izolării sau ocolire a aprobărilor promise este criteriu eliminativ, chiar dacă demo-ul arată bine.

22. Implementare etapizată și alternative

Plan de livrare

Etapa 0 — inventar și prototip de izolare. Confirmăm tema cPanel, API-urile, versiunile, resursele și mecanismul CageFS/LVE. Demonstrăm proces per user, cleanup, lipsa accesului transversal și separarea live dorită. Fără acest rezultat nu lansăm un agent cu write pentru clienți.

Etapa 1 — chat comun și sysadmin read-only. Implementăm autentificarea, joburile, un engine, catalogul minim de diagnostic și rapoarte verificabile. Include escaladare către suport, nu remedieri root.

Etapa 2 — reparații și builder controlat. Introducem workspace, teste, preview, ChangeSet și deployer pentru un set restrâns de aplicații. Sysadminul folosește manifestele acestora și poate iniția reparații pe același traseu.

Etapa 3 — Commerce Assistant. Adăugăm pluginul HostGate, abilities, tema/blocurile susținute și fluxuri pentru produse/comenzi. Validăm explicit publicarea fără pierderea datelor live și integrările comerciale.

Etapa 4 — comparație OpenCode și extindere. Aceeași fundație primește un al doilea adaptor; alegerea engine/model se face după benchmark. Creștem concurența și catalogul de operațiuni numai după testele de regresie și cost.

Ordinea nu presupune că WooCommerce este mai puțin important comercial; urmărește construirea fundației o singură dată. Etapele pot fi suprapuse de echipe diferite, dar barierele de securitate și contractele comune nu se ocolesc.

Alternative relevante

Codex App Server prin STUDIO: candidat pentru interacțiuni bidirecționale și aprobări integrate în timpul execuției. Nu cere HTTP comun. Maturitatea interfeței și traseul de SDK ales trebuie verificate pe versiunea fixată; nu presupunem că un avertisment despre un transport se aplică identic tuturor pachetelor. [S19]

Claude Agent SDK: alternativă pentru reutilizarea buclei și instrumentelor Claude Code când modelul/engine-ul Claude este alegerea deliberată. Necesită aceleași controale HostGate; verificăm autentificarea comercială permisă de provider. [S32]

ACP: poate standardiza comunicarea dintre chat/backend și engine-uri compatibile. Nu înlocuiește autorizarea, sandboxul sau Abilities. Îl evaluăm dacă funcțiile comune și adaptoarele existente reduc integrarea reală, nu doar pentru încă un standard în schemă. [S33]

Workers de dezvoltare separați de hosting: evoluție utilă când buildurile sunt grele ori este dificilă izolarea strictă lângă live. Agentul rulează într-un mediu temporar pe altă infrastructură, iar pe cPanel rămân collectorul de cont și deployerul. OpenHands SDK este un candidat pentru evaluarea unor medii de agent dedicate, nu o condiție a primei versiuni. [S34]

Această alternativă nu abandonează ideea per client și la cerere. Schimbă locul execuției și canalul de acces la proiect, menținând faptul că niciun agent nu are acces global nefiltrat.

23. Decizii pentru review și concluzie

Ce trebuie aprobat înainte de implementare

Decizie	Recomandare inițială
Produs	Un chat cPanel dev + sysadmin; Commerce pe același backend
Engine inițial	CLI în backend PHP sau SDK TS în worker Node
Izolare promisă	Nivel comercial controlat, nu doar prompt read-only
Scope sysadmin	Cont/proiect; fără root și fără servicii globale
Scope App Builder	Catalog restrâns de runtime-uri și templates testate
Scope WooCommerce	Catalog de operațiuni și integrări susținute
Publicare	Plan/artefact verificat + aprobare exactă + executor
Model comercial LLM	API/gateway cu buget și tokenuri limitate
Scalare	Procese la cerere, limite măsurate, coadă echitabilă
Compatibilitate cPanel nou	Test explicit Jupiter/Meridian, fără presupuneri

Verdict arhitectural

Ideea poate fi extinsă de la WooCommerce la sysadmin și App Builder fără să reconstruiești agentul.

Sysadminul și builderul sunt chiar complementare: unul înțelege și verifică mediul aplicației, celălalt produce schimbările. Un chat comun este o alegere coerentă pentru utilizator.

WooCommerce rămâne special prin modelul de date, instrumentele WordPress și efectele comerciale. Nu îl izolăm într-o platformă complet diferită, dar nici nu tratăm comenzile și stocul ca simple fișiere de proiect.

Investiția HostGate ar trebui concentrată în identitate și izolare, tools de cont, workspace/deployer, Abilities, testare și experiența de aprobare. CLI versus SDK rămâne o decizie de integrare. Codex versus OpenCode este o decizie de compatibilitate și rezultate măsurate.

Formula recomandată: un chat + o platformă de control + agent existent la cerere + profiluri și executori separați. Nici agent root pentru toți clienții, nici trei aplicații rescrise și nici o orchestrare cu mai multe LLM-uri înainte ca fluxul simplu să fie validat.

Anexa A. Manifest de compatibilitate pentru pilot

Manifestul se completează cu valori reale, nu cu „latest”. Se păstrează împreună cu rezultatele testelor și se actualizează la fiecare rollout.

```
# Schema propusa HostGate; valorile trebuie completate in pilot.
platform:
  os_release: TO_VALIDATE
  kernel: TO_VALIDATE
  cpanel_version: TO_VALIDATE
  cpanel_theme: TO_VALIDATE
  cloudlinux_cagefs_lve: TO_VALIDATE
  isolation_profile: TO_VALIDATE
engines:
  codex_cli_version_and_checksum: TO_VALIDATE
  codex_sdk_ts_version: TO_VALIDATE
  node_worker_version: TO_VALIDATE
  opencode_version_and_cli_generation: TO_VALIDATE
model_access:
  provider_and_model_ids: TO_VALIDATE
  gateway_protocol_tests: TO_VALIDATE
  budget_policy_version: TO_VALIDATE
applications:
  supported_runtime_templates: TO_VALIDATE
  deployer_and_verifier_versions: TO_VALIDATE
wordpress:
  php_cli_and_web: TO_VALIDATE
  wp_cli: TO_VALIDATE
  wordpress: TO_VALIDATE
  woocommerce: TO_VALIDATE
  hostgate_plugin_and_theme: TO_VALIDATE
  mcp_adapter_if_enabled: TO_VALIDATE
acceptance:
  isolation_suite: NOT_RUN
  resume_cancel_retry_suite: NOT_RUN
  publication_and_recovery_suite: NOT_RUN
  benchmark_report: NOT_RUN
```

Acest manifest nu recomandă o versiune Node sau PHP doar pentru că apare în documentația unei componente. Se alege combinația susținută de hosting și securitate și se verifică integrările. Schimbarea unui engine sau a unui provider poate necesita rerularea testelor de tool calling, autentificare și evenimente.

Anexa B. Checklist operațional per job

Înainte de pornire: actorul și proiectul sunt autorizate; contul este activ; engine-ul și profilul sunt aprobate; nu există conflict de modificare; bugetul este disponibil; calea/workspace-ul sunt rezolvate din inventar; tokenul de job este emis; contextul nu conține secrete inutile.

După pornire: UID/GID și izolarea sunt cele așteptate; sunt vizibile doar resursele permise; se înregistrează procesul/cgroup-ul; se citesc stdout și stderr fără blocare; evenimentele sunt limitate și filtrate; progresul nu este confundat cu succesul.

La aplicare: aprobarea este server-side, neexpirată și pentru planul exact; hash-ul și starea de bază corespund; lock-ul și fencing-ul sunt valide; efectele sunt jurnalizate; codul clientului nu se execută cu drepturile infrastructurii.

La terminare: rezultatul este verificat; se reconciliază operațiunile parțiale; se revocă accesul temporar; se opresc procesele nepreluate de runtime; se curăță preview-ul conform retenției; se salvează sesiunea și auditul; clientul vede exact ce s-a făcut și ce rămâne neconfirmat.

La escaladare: echipa primește resursa, dovezile, versiunea, pașii executați și efectele posibile. Nu primește o concluzie certă dacă diagnosticul este doar probabil. Nu se atașează nefiltrat parole, loguri ale altor conturi sau date comerciale inutile.

Anexa C. Surse primare și limite ale verificării

Sursele de mai jos susțin capabilitățile tehnice ale produselor. Schemele, numele modulelor HostGate, fluxurile și recomandările sunt propuneri de proiectare. Consultare: **16 septembrie 2026**. Unele URL-uri OpenAI redirectionează către documentația oficială ChatGPT Learn.

Sursele nu reprezintă teste pe infrastructura HostGate. Ramurile de cod main și dev se pot modifica. Înainte de implementare se fixează versiunile și se verifică manifestul din Anexa A. Referințele cu mai multe numere indică un grup de surse relevante, nu existența unei singure funcții universale.

S01 — cPanel — Guide to cPanel Plugins: Add Plugins.

<https://api.docs.cpanel.net/guides/guide-to-cpanel-plugins/guide-to-cpanel-plugins-add-plugins/>

S02 — cPanel — Create a New cPanel Interface in PHP.

<https://api.docs.cpanel.net/guides/quickstart-development-guide/tutorial-create-a-new-cpanel-interface-in-php>

S03 — cPanel — Introduction to UAPI.

<https://api.docs.cpanel.net/cpanel/introduction/>

S04 — cPanel — Guide to API Authentication.

<https://api.docs.cpanel.net/guides/guide-to-api-authentication/>

S05 — cPanel — Guide to the LiveAPI System.

<https://api.docs.cpanel.net/guides/guide-to-the-liveapi-system/>

S06 — cPanel — Application Manager.

<https://docs.cpanel.net/cpanel/software/application-manager/>

S07 — cPanel — Meridian Release Notes.

<https://docs.cpanel.net/release-notes/meridian-release-notes/>

S08 — CloudLinux — Command-line tools; Running command inside CageFS.

https://docs.cloudlinux.com/cloudlinuxos/command-line_tools/

S09 — Docker — Docker Engine security.

<https://docs.docker.com/engine/security/>

S10 — OpenAI — Codex non-interactive mode.

<https://developers.openai.com/codex/noninteractive/>

S11 — OpenAI — Codex SDK: TypeScript și Python.

<https://developers.openai.com/codex/sdk/>

S12 — OpenAI — Codex TypeScript SDK README.

<https://github.com/openai/codex/blob/main/sdk/typescript/README.md>

S13 — OpenAI — SDK exec.ts: lansarea procesului.

<https://raw.githubusercontent.com/openai/codex/main/sdk/typescript/src/exec.ts>

S14 — OpenAI — SDK codexOptions.ts: cale, mediu și configurare.

<https://raw.githubusercontent.com/openai/codex/main/sdk/typescript/src/codexOptions.ts>

S15 — OpenAI — SDK thread.ts: sesiuni, streaming și anulare.

<https://raw.githubusercontent.com/openai/codex/main/sdk/typescript/src/thread.ts>

S16 – OpenAI – SDK threadOptions.ts: sandbox și permisiuni.

<https://raw.githubusercontent.com/openai/codex/main/sdk/typescript/src/threadOptions.ts>

S17 – OpenAI – Advanced Configuration.

<https://developers.openai.com/codex/config-advanced/>

S18 – OpenAI – Sandbox.

<https://developers.openai.com/codex/sandboxing/>

S19 – OpenAI – Codex App Server.

<https://developers.openai.com/codex/app-server/>

S20 – OpenCode – CLI.

<https://opencode.ai/docs/cli/>

S21 – OpenCode – SECURITY.md.

<https://raw.githubusercontent.com/anomalyco/opencode/dev/SECURITY.md>

S22 – OpenCode – Config.

<https://opencode.ai/docs/config/>

S23 – OpenCode – Providers.

<https://opencode.ai/docs/providers/>

S24 – OpenCode – Permissions.

<https://opencode.ai/docs/permissions/>

S25 – OpenCode v2 – CLI Commands.

<https://opencode.ai/v2/docs/cli/commands/>

S26 – WordPress – wp ability run.

<https://developer.wordpress.org/cli/commands/ability/run/>

S27 – WordPress – wp_register_ability().

https://developer.wordpress.org/reference/functions/wp_register_ability/

S28 – WooCommerce – MCP Integration și abilities.

<https://developer.woocommerce.com/docs/features/mcp/>

S29 – WooCommerce – How to update WooCommerce.

<https://woocommerce.com/document/how-to-update-woocommerce/>

S30 – WordPress – Abilities API: Getting started.

<https://developer.wordpress.org/apis/abilities-api/getting-started/>

S31 – WordPress – WP_Ability.

https://developer.wordpress.org/reference/classes/wp_ability/

S32 – Anthropic – Claude Agent SDK overview.

<https://code.claude.com/docs/en/agent-sdk/overview>

S33 – Agent Client Protocol – Introduction.

<https://agentclientprotocol.com/get-started/introduction>

S34 – OpenHands – Software Agent SDK.

<https://docs.openhands.dev/sdk>

S35 – OpenAI – Authentication.

<https://developers.openai.com/codex/auth/>

S36 – cPanel & WHM – Developer Portal; rolurile cPanel și WHM.

<https://api.docs.cpanel.net/>

S37 – WordPress – wp eval.

<https://developer.wordpress.org/cli/commands/eval/>

S38 – WordPress – wp ability.

<https://developer.wordpress.org/cli/commands/ability/>

S39 – OpenCode – Troubleshooting și locațiile datelor.

<https://opencode.ai/docs/troubleshooting/>