ARCHITEKTUR-FALLSTUDIE 7D / 01

Fernsteuerbare Entwicklungsinfrastruktur

Eine Windows-basierte Remote-Architektur für autorisierten Werkzeugzugriff, Startstabilität und lesende Betriebsdiagnose.

Ausgangslage und Ziel

KI-gestützte Entwicklungsarbeit sollte von einem mobilen Gerät aus möglich sein, ohne den PC jedes Mal lokal bedienen oder Desktop Commander manuell starten zu müssen. Gleichzeitig mussten Windows-Anmelderegeln, Zugriffsrechte und produktive Projektzustände unangetastet bleiben.

Ziel waren ein robuster Verbindungsaufbau nach einem echten Neustart, klar getrennte Statusdiagnose und bestehende Sicherheitsgrenzen ohne neue Netzwerkdienste.

Lösungsarchitektur

Der Windows-11-Pro-Rechner stellt die autorisierte Ausführungsumgebung bereit. Ein bestehender Boot-Task startet einen intelligenten Netzwerk-Wächter, der die Verbindung nur nach Netzbereitschaft herstellt. Desktop Commander liefert die Remote-Brücke; PowerShell führt projektgebundene Diagnosen aus. VPN/RDP ergänzen den interaktiven Zugriff, ohne die Commander-Architektur zu ersetzen.

Ein lesender Statuskern beurteilt Haupt- und Kindprozess, Dienst-/Task-Zustand, Netzwerk und Erreichbarkeit. Änderungen oder Selbstreparaturen führt dieser Statuskern nicht aus.

KOMPONENTEN- UND VERANTWORTUNGSFLUSS

Windows-Boot	Netzwerk bereit	Remote-Agent	Lesende Diagnose

Entwurfsprinzip

Trennung von Entscheidung, Ausführung und Abnahme. Die Architektur ist gezielt begrenzt: Sie bewahrt bestehende Systeme und macht Verantwortlichkeit überprüfbar, statt maximale Autonomie ungeprüft zu behaupten.

Realisierung und Integration

Ausgewählte technische Schritte, Schnittstellen und Entscheidungen – ohne unnötige Implementierungsdetails aus geschützten Systemen.

Technische Umsetzung

Die Boot-Architektur vermeidet eine starre 60-Sekunden-Pause und überprüft die tatsächliche Netzbereitschaft mit begrenzten Wiederholungen. Das System startet weder einen zweiten Commander noch benötigt es eine automatische Windows-Anmeldung.

Im realen Test startete Windows um 01:36:23, das Netz war um 01:36:49 bereit und der Commander um 01:36:56 online. Anschließend wurde ein Remote-PowerShell-Befehl ausgeführt, obwohl kein interaktiver Windows-Benutzer angemeldet war.

Qualitätssicherung und Abnahmepunkte

Der isolierte Diagnosekern bestand 17/17 Tests: 15 synthetische Statusfälle und zwei statische Sicherheitsprüfungen. Ein Live- und JSON-Test bestätigte den lesenden Modus, die erwartete Prozessstruktur und null ausgeführte Änderungen.

Die Bootsicherung und das Ereignisprotokoll belegen die konkrete Startsequenz. Der nachgewiesene Commander-Onlinezustand lag 33 Sekunden nach Windows-Boot; diese Messung gilt für den dokumentierten Neustart.

Warum diese Architektur?

Sie unterstützt inkrementelle Umsetzung mit kleinen, beobachtbaren Änderungen. Jede Stufe besitzt ein konkretes Prüfkriterium. Fehler oder Grenzen werden dokumentiert, nicht durch eine positive Statusmeldung überdeckt.

Prüfstrategie

- Baseline sichern und den Umfang einer Änderung begrenzen.
- Den tatsächlich verwendeten Integrationspfad statt nur einer isolierten Komponente kontrollieren.
- Fehlerfälle und unerlaubte Zustände bewusst testen.
- Ergebnis, Restunsicherheiten und Wiederaufnahme reproduzierbar dokumentieren.

Belege, Risiken und Lernwert

Ein belastbarer Referenzbericht unterscheidet echte Beobachtung, isolierten Test und noch nicht erreichte Betriebsziele.

Sicherheitsgrenzen und offene Punkte

Keine zusätzlichen Netzwerkports, keine neuen Privilegien und kein AutoAdminLogon. Genau ein Commander-Hauptprozess mit zugeordnetem Kindprozess, kein zusätzlicher Autostart. Das System trennt Diagnose und Veränderung.

Offen sind die komplette Smart-Home-Kaltstartkette nach Stromtrennung, langfristige Stabilität bei jeder Netzstörung und eine generelle Garantie für GUI-Interaktion ohne Benutzeranmeldung.

Wirkung und eigener Beitrag

Die Fallstudie zeigt Infrastrukturarchitektur jenseits einzelner Skripte: zeitabhängige Boot-Reihenfolge, Zustandsbeobachtung, Rechtebegrenzung, reale Messung und reproduzierbare Abnahme.

Mein Beitrag: Zieldefinition, Architektur- und Sicherheitsentscheidungen, Auswahl der Wiederholstrategie, Testfreigabe, Messkriterien und unabhängige Ergebnisprüfung; Implementierung und Instrumentierung wurden KI-gestützt koordiniert.

Übertragbarkeit

Das Vorgehen ist auf AI-Integration, interne Automatisierung und Developer-Experience-Plattformen übertragbar: klare Schnittstellen, minimale Rechte, dokumentierte Zustände, Kostenbewusstsein und reproduzierbare Kontrollen. Ein neues Zielsystem erfordert selbstverständlich eine eigene Sicherheits- und Abnahmeprüfung.

Lessons learned

Ein aussagekräftiger Architektur-Nachweis zeigt nicht nur einen erfolgreichen Endzustand. Er macht auch den Weg zur Fehlerentdeckung, die Grenzen des Tests und die Verantwortung für reale Freigaben sichtbar. Die Dokumentation ist deshalb Teil des technischen Ergebnisses und kein nachträgliches Marketingmaterial.

Quellenbasis: interne Projekthaltepunkte vom 22.–25.11.2026, technische Testprotokolle und Portfolio-Register. In dieser Bewerbungsfassung bewusst ohne interne Pfade, Tokens, private Repository-URLs oder personenbezogene Systemdaten.