ARCHITEKTUR-FALLSTUDIE 7C / 01

Autonome TangoKunstWerk-Werkbank

Eine WordPress-nahe Entwicklungsarchitektur für Kursverwaltung, sichere Vorschauen und nachweisbare Qualitätskontrollen.

Ausgangslage und Ziel

Wiederkehrende Kurstermine und Ausfälle mussten nachvollziehbar über mehrere Standorte hinweg gepflegt werden. Unterschiedliche Bedienoberflächen und parallele Datenstände hätten fehlerhafte Termine oder unkontrollierte Veröffentlichungen begünstigt.

Ziel war eine einzige fachliche Datenbasis, eine verständliche mobile Administration und ein klar getrenntes Modell für Vorschau, Freigabe und Veröffentlichung.

Lösungsarchitektur

WordPress bildet die produktive Datenbasis. Modern und Klassisch sind alternative Oberflächen für dieselben Kursoptionen. Der Kalenderassistent validiert Zeiträume, zeigt betroffene Termine und fordert eine explizite Bestätigung vor dem Publish. PowerShell-Gateways und spezialisierte MCP-/Browserwerkzeuge dienen der technischen Koordination und Prüfung.

Im Geltungsbereich liegen Saarbrücken, Speyer und Bad Dürkheim. Homburg ist bewusst von dieser Termin-Automatik ausgeschlossen.

KOMPONENTEN- UND VERANTWORTUNGSFLUSS

Dateneingabe	Validierung	Echte WP-Vorschau	Explizite Freigabe

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 Bedienoberfläche akzeptiert mehrere eindeutige Datumsformate, erkennt ungültige Eingaben und erlaubt die Standortauswahl. Die Integration verwendet keine parallele zweite Kursdatenbank und keinen zweiten schreibenden Prozess.

Bei einer Integrationsprüfung des tatsächlich eingesetzten PHP-Bundles wurde eine fehlende Vergleichsklasse erkannt und gezielt einmalig ergänzt. Das war ein wesentlicher Qualitätsgewinn gegenüber einer bloß visuell erfolgreich geladenen Administrationsseite.

Qualitätssicherung und Abnahmepunkte

Nachgewiesen sind 26 PHP-Datumstests, 18 Browserfälle und 28+11 Realcode-Tests. Ein echter autorisierter WordPress-Preview-POST ermittelte eine Änderung, ohne Publish-POST; fünf Kursoptionen blieben unverändert.

Modern/Klassisch wurden in 14/14 lesenden Live-Adminansichten geprüft. Der Betreiber bestätigte den Kalenderassistenten persönlich in iPhone-Safari. Außerdem bestanden 21/21 isolierte Controller-HTTP-Tests mit WordPress-Stubs zu Vorschau, Einmaltoken, Publish und Restore.

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.

ARCHITEKTUR-FALLSTUDIE 7C / 03

Belege, Risiken und Lernwert

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

Sicherheitsgrenzen und offene Punkte

Publikation verlangt eine konkrete Freigabe; Voransicht ist ausdrücklich keine Veröffentlichung. Negativtests, SHA-Rücklesung, Revisions- und Protokollkontrollen sowie Schutz unbetroffener Optionen sind Bestandteil des Verfahrens.

Offen bleibt die erste echte produktive Veröffentlichung einer Abwesenheit einschließlich Live-Readback. Die 21 isolierten Restore-Prüfungen sind kein vollständiger WordPress-Server-Restore.

Wirkung und eigener Beitrag

Die Lösung verbindet UX, Backend-State, Sicherheit und testgetriebene Integration. Auch technische Rückschläge – wie die fehlende Klasse – wurden nicht kaschiert, sondern am realen Bundle nachgewiesen und korrigiert.

Mein Beitrag: Anforderungen, Ausfall- und Kurslogik, Usability-Abnahme, technische Arbeitskoordination, Freigabegrenzen und unabhängige Qualitätsprüfung; die Implementierung erfolgte KI-gestützt.

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.