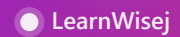


Enterprise-Architektur mit Wisej.NET, technische Führung und Governance

Lektionsleitfaden · Lab- und Prüfungsleitfaden



Lektionsleitfaden

Architekturentscheidungen, Projektstandards, Team-Workflow, Code-Review-Gates. Was das für ein Wisej.NET-Team auf Expertenniveau bedeutet – und wie Sie dieses Modul lesen.



Fokus: Produktivbetrieb

Architekturentscheidungen, Projektstandards, Team-Workflow, Code-Review-Gates.

1. Das Problem im Produktivbetrieb

Bei fortgeschrittener Arbeit mit Wisej.NET geht es nicht mehr darum, ob sich ein Bildschirm bauen lässt. Es geht darum, ob ein Team eine große Anwendung über Jahre weiterentwickeln kann, ohne dass Produktivität zu versteckten technischen Schulden wird.

2. Zentrale Konzepte

Konzept	Warum es wichtig ist
Enterprise-Lösungsstruktur	Teilt die Lösung in Projekte für UI, Anwendungsservices, Domäne und Infrastruktur auf, damit Bildschirme schlank bleiben und Geschäftsregeln genau einen Ort haben.
Architecture Decision Records	Kurze Dokumente, die Kontext, Optionen, Entscheidung und Konsequenzen einer Designentscheidung festhalten, damit das Team sie später prüfen und neu bewerten kann.
designerfreundliche Abgrenzungen	Halten Forms und Pages im Designer bearbeitbar, während Workflow, Validierung und Datenzugriff in Services liegen, die der Designer nie anfasst.
Review-Gates	Festgelegte Prüfpunkte, an denen Code, Architektur und Sicherheit geprüft werden, bevor ein Feature gemergt wird – so wird Qualität durchgesetzt, statt nur erhofft.
Teamkonventionen	Gemeinsame Regeln für Benennung, Ordner und Service-Registrierung, durch die jeder Bildschirm jedem Entwickler im Team vertraut vorkommt.
Kontrolle technischer Schulden	Praktiken wie Schuldenprotokolle, Refactoring-Budgets und Regeln für Deprecation, die Abkürzungen sichtbar und eingeplant halten, statt sie in Vergessenheit geraten zu lassen.

3. Der rote Faden der Lektion

Entwickler auf mittlerem Niveau können eine produktivorientierte Anwendung bauen. Experten entscheiden, wie die Anwendung ihr Wachstum übersteht: mehr Bildschirme, mehr Benutzer, mehr Integrationen, mehr Teams und mehr Deployment-Umgebungen. Das Ziel ist nicht, eine generische Webarchitektur nachzuahmen. Das Ziel ist, den Vorteil von Wisej.NET zu bewahren – hohe Produktivität mit serverseitigem C# – und gerade so viel Disziplin hinzuzufügen, dass Veränderungen im Enterprise-Umfeld tragfähig bleiben.

Governance ohne Bürokratie

Ein Wisej.NET-Team sollte kein Gremium brauchen, um einen Button hinzuzufügen. Es braucht aber Standards dafür, wo Workflow-Logik hingehört, wie Bildschirme Services nutzen, wie Sicherheit durchgesetzt wird, wie der Datenzugriff abstrahiert wird und wie Diagnose ergänzt wird. Schlanke Architecture Decision Records genügen, wenn sie Entscheidung, Kontext, Konsequenzen und Termin für die nächste Überprüfung erläutern.

Designer-first heißt nicht ohne Architektur

Der Visual Studio Designer bleibt auch in anspruchsvollen Projekten wertvoll. Die Grenze verläuft nicht zwischen visuell entworfener und programmierter UI, sondern zwischen UI-Zustand und Anwendungsentscheidungen. Forms, Pages und UserControls können im Designer bearbeitbar bleiben, während Orchestrierung, Autorisierung, Validierung, Persistenz und Integration in Services bleiben.



Expertenregel für den Produktivbetrieb

Hören Sie nicht auf, sobald der Bildschirm korrekt aussieht. Erklären Sie, wem der Zustand gehört, welche Sicherheitsfolgen es gibt, welche Fehlerpfade existieren und wie sich das System im Produktivbetrieb verhält, bevor Sie das Modul als abgeschlossen betrachten.



Lernziele

Am Ende dieses Moduls können Sie:

- ✓ Architekturgrenzen für eine große Wisej.NET-Anwendung definieren, ohne sie zu überfrachten.
- ✓ Projektstandards für UI, Services, Datenzugriff, Integrationen, Sicherheit, Diagnose und Deployment erstellen.
- ✓ Architecture Decision Records und Review-Gates nutzen, um Teams auf einer Linie zu halten.
- ✓ erklären, wie die Produktivität von Wisej.NET die Verantwortung eines Technical Lead verändert.

Lab- und Prüfungsleitfaden

Was Sie im Praxis-Lab bauen, das vorgeschlagene Vorgehen und die geforderten Ergebnisse.



Praxis-Lab

Legen Sie die Basis der Lösung EnterpriseOps Command Center an. Fügen Sie Projekte oder Ordner für UI, gemeinsame Controls, Domäne, Anwendungsservices, Datenzugriff, Integrationen, Sicherheit, Diagnose, Ressourcen und Deployment hinzu. Ergänzen Sie einen Ordner für Architecture Decision Records und schreiben Sie ADR-001 zur gewählten Lösungsstruktur.

Vorgeschlagenes Vorgehen

- ✓ Lesen Sie das Problem im Produktivbetrieb und benennen Sie das Architekturrisiko, bevor Sie die UI anfassen.
- ✓ Legen Sie zuerst die betroffene Grenze für Service, Command, Modell oder Komponente an oder aktualisieren Sie sie.
- ✓ Erstellen Sie den Wisej.NET-Bildschirm oder die Komponente nach den etablierten Projektstandards.
- ✓ Demonstrieren Sie auch einen Fehlerpfad, nicht nur den Happy Path.
- ✓ Halten Sie die Änderungen im Lernprotokoll fest und aktualisieren Sie die zugehörige Checkliste.

Geforderte Ergebnisse

- ✓ Screenshot oder Baumansicht der Lösungs- und Ordnerstruktur
- ✓ ADR-001 mit der Entscheidung zur Lösungsstruktur
- ✓ Seite mit den Coding-Standards des Teams
- ✓ Beispiel für die Benennung eines Referenzbildschirms und für eine Service-Kapselung
- ✓ Auf den ersten Bildschirm angewendete Code-Review-Checkliste

Wiederholungsfragen

- ✓ Findet ein neuer Senior-Entwickler den Workflow-Service zu einem Bildschirm in weniger als einer Minute?
- ✓ Kann das Team erklären, warum eine Grenze existiert?
- ✓ Ist der Designer nach der Aufteilung der Architektur noch nutzbar?



Expertenregel für den Produktivbetrieb

Hören Sie nicht auf, sobald der Bildschirm korrekt aussieht. Erklären Sie, wem der Zustand gehört, welche Sicherheitsfolgen es gibt, welche Fehlerpfade existieren und wie sich das System im Produktivbetrieb verhält, bevor Sie das Modul als abgeschlossen betrachten.