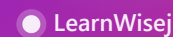


Enterprise Wisej.NET Architecture, Technical Leadership, and Governance

Lesson Guide · Lab / Exam Guide



Lesson Guide

Architecture decisions, project standards, team workflow, code review gates. What this means for an advanced Wisej.NET team — and how to read this module.



Production focus

Architecture decisions, project standards, team workflow, code review gates.

1. The production problem

Advanced Wisej.NET work is no longer about whether a screen can be built. It is about whether a team can evolve a large application for years without turning productivity into hidden technical debt.

2. Key concepts

Concept	Why it matters
enterprise solution structure	Splits the solution into UI, application services, domain, and infrastructure projects so screens stay thin and business rules have one home.
architecture decision records	Short documents that capture the context, options, decision, and consequences of a design choice so the team can review and revisit it later.
designer-friendly boundaries	Keeps forms and pages designer-editable while routing workflow, validation, and data access into services the designer never touches.
review gates	Defined checkpoints where code, architecture, and security are reviewed before a feature merges, so quality is enforced rather than hoped for.
team conventions	Shared naming, folder, and service-registration rules that make every screen look familiar to any developer on the team.
technical debt controls	Practices such as debt logs, refactoring budgets, and deprecation rules that keep shortcuts visible and scheduled instead of forgotten.

3. Lesson narrative

Intermediate developers can build a production-oriented application. Advanced developers decide how the application survives growth: more screens, more users, more integrations, more teams, and more

deployment environments. The goal is not to imitate a generic web architecture. The goal is to preserve the Wisej.NET advantage - rich server-side C# productivity - while adding enough discipline to support enterprise change.

Governance without bureaucracy

A Wisej.NET team should not need a committee to add a button. But it does need standards for where workflow logic goes, how screens use services, how security is enforced, how data access is abstracted, and how diagnostics are added. Lightweight architecture decision records are enough when they explain the decision, context, consequences, and review date.

Designer-first does not mean architecture-free

The Visual Studio designer remains valuable even in advanced projects. The boundary is not visual versus coded UI; the boundary is UI state versus application decisions. Forms, Pages, and UserControls can remain designable while orchestration, authorization, validation, persistence, and integration stay in services.



Advanced production habit

Do not stop when the screen looks correct. Explain state ownership, security implications, failure paths, and production behavior before considering the module complete.



Learning objectives

By the end of this module you'll be able to:

- ✓ Define architecture boundaries for a large Wisej.NET application without over-engineering it.
- ✓ Create project standards for UI, services, data access, integrations, security, diagnostics, and deployment.
- ✓ Use architecture decision records and review gates to keep teams aligned.
- ✓ Explain how Wisej.NET productivity changes the responsibilities of a technical lead.

Lab / Exam Guide

What you'll build in the hands-on lab, the suggested approach, and the required deliverables.



Hands-on lab

Create the EnterpriseOps Command Center solution baseline. Add projects or folders for UI, shared controls, domain, application services, data access, integrations, security, diagnostics, resources, and deployment. Add an Architecture Decision Record folder and write ADR-001 for the chosen solution structure.

Suggested approach

- ✓ Read the production problem and identify the architecture risk before touching the UI.
- ✓ Create or update the relevant service, command, model, or component boundary first.
- ✓ Build the Wisej.NET screen or component using the established project standards.
- ✓ Add a failure-path demonstration, not only the happy path.
- ✓ Record what changed in the learning log and update the relevant checklist.

Required deliverables

- ✓ Solution/folder structure screenshot or tree
- ✓ ADR-001 solution structure decision
- ✓ Team coding standards page
- ✓ Reference screen naming and service-boundary example
- ✓ Code review checklist applied to the first screen

Student review questions

- ✓ Can a new senior developer find the workflow service for a screen in under one minute?
- ✓ Can the team explain why a boundary exists?
- ✓ Is the designer still usable after the architecture split?



Advanced production habit

Do not stop when the screen looks correct. Explain state ownership, security implications, failure paths, and production behavior before considering the module complete.