Natech
BANKING SOLUTIONS

PROJECT FUTURE · CURRICULUM

AI Enabled Application Development Practices

Εκπαιδευτική Ακαδημία Application Development powered by Natech Banking Solutions

ΔΙΑΡΚΕΙΑ
40 ώρες · 8 modules

ΗΜΕΡΟΜΗΝΙΕΣ
25/1/2027 – 10/3/2027

ΠΡΟΓΡΑΜΜΑ
Δευτέρα · Τετάρτη · Παρασκευή
15:00–17:00

Επισκόπηση

Στο τέλος του προγράμματος, ο συμμετέχων γράφει, ελέγχει και παραδίδει κώδικα C# με τη βοήθεια AI εργαλείων με υπεύθυνο τρόπο, και χτίζει ένα ASP.NET Core Web API που χρησιμοποιεί LLM.

Γλώσσα και εργαλεία

C# και .NET 10 (LTS), Visual Studio ή VS Code με C# Dev Kit, Git/GitHub και GitHub Actions, ένας AI coding assistant μέσα στο IDE, ένα agentic coding εργαλείο (π.χ. Claude Code), ένα chat assistant και ένα LLM API μέσω του .NET SDK του παρόχου ή του Microsoft.Extensions.AI. Η C# δίνει συνέχεια με το «Mobile Development with C#»· οι αρχές είναι ανεξάρτητες από γλώσσα και εργαλείο.

Running project: Smart Notes

Μία εφαρμογή εξελίσσεται από module σε module: το Smart Notes, μια εφαρμογή σημειώσεων που ξεκινά ως console app, γίνεται ASP.NET Core Web API και καταλήγει να απαντά σε ερωτήσεις πάνω στις σημειώσεις του χρήστη.

Πλαίσιο AI-native SDLC

Το πρόγραμμα δανείζεται, απλοποιημένες για αρχάριους, ιδέες από το «AI-native SDLC playbook» της Anthropic (Claude Academy). Ο κύκλος χτίζεται σταδιακά από το Module 2 και ολοκληρώνεται στο Module 8.

intent.md → spec → tests → AI & ανθρώπινο review → evals → παραγωγή

↺ ό,τι εντοπίζεται στην παραγωγή γίνεται νέο intent.md

MODULE	ΘΕΜΑ	ΤΙ ΠΡΟΣΤΙΘΕΤΑΙ ΣΤΟ SMART NOTES	ΔΙΑΡΚΕΙΑ
1	**AI Fundamentals για developers**	Setup .NET, πρώτο console app	**4 ώρες**
2	**Prompting για κώδικα**	Κλάση Note, δημιουργία / αναζήτηση / διαγραφή, πρώτο intent.md	**4 ώρες**
3	**Testing και debugging με AI**	xUnit tests, διόρθωση bugs, αποθήκευση σε JSON	**6 ώρες**
4	**Git, code review και ομαδική δουλειά**	Repo, CLAUDE.md, pull requests, CI με GitHub Actions	**4 ώρες**
5	**Ασφάλεια, απόρρητο, κανονιστικό πλαίσιο**	Security review, user secrets, hooks ως approval gates	**4 ώρες**
6	**Εφαρμογές με LLM APIs**	Web API, περίληψη και κατηγοριοποίηση με LLM	**6 ώρες**
7	**RAG, tool use, agents**	Ερωτήσεις πάνω στις σημειώσεις, tool «δημιούργησε σημείωση»	**6 ώρες**
8	**Evaluation και παραγωγή**	Evals στο CI, monitoring, τελικό project	**6 ώρες**
	Σύνολο		**40 ώρες**

Αξιολόγηση: κάθε άσκηση παραδίδεται στον εισηγητή (Modules 1–3 με email, από το Module 4 με pull request). Το τελικό project παρουσιάζεται στο Module 8.

MODULE 1 · 4 ΩΡΕΣ

AI Fundamentals για developers

Ο συμμετέχων καταλαβαίνει τι κάνει ένα LLM, πού κάνει λάθη και στήνει το περιβάλλον εργασίας του.

1.1	**Ιστορία και ορισμοί:** AI, machine learning, LLM, generative AI. Πώς φτάσαμε από τα πρώτα chatbots στους σημερινούς coding assistants.
1.2	**Πώς λειτουργεί ένα LLM:** Tokens, πρόβλεψη επόμενης λέξης, context window, training cutoff. Γιατί το ίδιο prompt δίνει διαφορετικές απαντήσεις.
1.3	**Όρια και λάθη:** Hallucinations, ξεπερασμένη γνώση για βιβλιοθήκες, «σίγουρος τόνος» χωρίς σιγουριά. Παράδειγμα: ζητάμε κώδικα για ανύπαρκτη συνάρτηση.
1.4	**Κατηγορίες εργαλείων:** Chat assistants, assistants μέσα στο IDE (autocomplete, inline chat), agentic εργαλεία που εκτελούν εντολές και αλλάζουν αρχεία.
1.5	**Setup περιβάλλοντος:** .NET SDK, Visual Studio ή VS Code με C# Dev Kit, Git, AI extension, λογαριασμός GitHub.
1.6	**Το πρώτο μου πρόγραμμα με AI:** Hello World σε C# (dotnet new console), γραμμένο με βοήθεια assistant, και εξήγηση γραμμή-γραμμή.

Άσκηση 1.7 — 30 λεπτά

Με τη βοήθεια AI, φτιάξτε ένα console app σε C# που εμφανίζει μενού με επιλογές «Νέα σημείωση», «Λίστα σημειώσεων» και «0 – Έξοδος». Οι σημειώσεις κρατιούνται σε List<string> στη μνήμη. Παραδώστε τον κώδικα και ένα σύντομο κείμενο που εξηγεί με δικά σας λόγια τι κάνει κάθε κομμάτι.

Άσκηση 1.8 — 30 λεπτά

Ζητήστε από δύο διαφορετικά εργαλεία την ίδια λύση. Καταγράψτε τρεις διαφορές και ένα λάθος ή μια υπόθεση που έκανε κάποιο από τα δύο.

MODULE 2 · 4 ΩΡΕΣ

Prompting για κώδικα

Ο συμμετέχων περιγράφει σωστά τι θέλει και διαβάζει κριτικά ό,τι του δίνει το AI.

2.1	**Από την ιδέα στο intent και το spec:** Πριν το prompt, γράφουμε ένα σύντομο intent.md (τι θέλουμε, γιατί, με ποιους περιορισμούς) και από αυτό ένα spec με inputs, outputs και κριτήρια αποδοχής. Και τα δύο μπαίνουν στο repo μαζί με τον κώδικα.
2.2	**Ανατομία ενός καλού prompt:** Ρόλος, context (γλώσσα, έκδοση .NET, υπάρχων κώδικας), συγκεκριμένο ζητούμενο, παραδείγματα, μορφή απάντησης.

2.3 **Δουλεύοντας σε βήματα:** Μικρά κομμάτια αντί για «φτιάξε μου όλη την εφαρμογή». Ζητάμε πρώτα σχέδιο (plan mode), το ελέγχουμε και μετά ζητάμε κώδικα.

2.4 **Context management:** Ποια αρχεία δίνουμε στο εργαλείο, πότε ξεκινάμε νέα συζήτηση, project instructions ως αρχείο στο repo (π.χ. CLAUDE.md) με κανόνες κώδικα και τις εντολές dotnet build και dotnet test.

2.5 **Διαβάζοντας AI-generated κώδικα:** Ζητάμε εξήγηση, εντοπίζουμε περιττή πολυπλοκότητα, ελέγχουμε αν τα NuGet packages και τα APIs που χρησιμοποιεί υπάρχουν πραγματικά.

2.6 **Πότε να μην χρησιμοποιούμε AI:** Όταν μαθαίνουμε μια βασική έννοια, όταν δεν μπορούμε να ελέγξουμε το αποτέλεσμα.

Άσκηση 2.7 45 λεπτά

Επεκτείνετε το Smart Notes ώστε κάθε σημείωση να είναι αντικείμενο τύπου Note (class ή record) με Title, Body και CreatedAt. Προσθέστε αναζήτηση με λέξη-κλειδί (LINQ) και διαγραφή. Υποχρεωτικά: γράψτε πρώτα intent.md και spec, παραδώστε τα prompts που χρησιμοποιήσατε και σημειώστε σε ποιο σημείο αλλάξατε τον κώδικα του AI και γιατί.

Άσκηση 2.8 30 λεπτά

Σας δίνεται ένα κομμάτι κώδικα γραμμένο από AI με τρία κρυμμένα προβλήματα. Βρείτε τα χωρίς να το εκτελέσετε.

MODULE 3 · 6 ΩΡΕΣ

Testing και debugging με AI

Ο συμμετέχων αποδεικνύει ότι ο κώδικας δουλεύει, αντί να το υποθέτει.

3.1 **Γιατί το testing μετράει περισσότερο με AI:** Γράφουμε πιο γρήγορα, άρα πρέπει να ελέγχουμε πιο συστηματικά.

3.2 **Unit tests με xUnit:** Δομή ενός test (Arrange–Act–Assert), [Fact] και [Theory], assertions, edge cases.

3.3 **AI για tests:** Ζητάμε tests για υπάρχοντα κώδικα και ελέγχουμε αν ελέγχουν κάτι ουσιαστικό ή απλώς «περνάνε».

3.4 **Test-first με AI:** Γράφουμε πρώτα τα tests από το spec, μετά ζητάμε υλοποίηση που τα περνά. Τα αποτελέσματα του dotnet test γίνονται feedback loop για το AI εργαλείο.

3.5 **Debugging:** Διαβάζοντας ένα exception και stack trace, breakpoints και debugger στο Visual Studio ή VS Code, πώς δίνουμε στο AI το σωστό context για ένα bug.

3.6 **Exceptions και error handling:** try/catch, δικά μας exceptions, αποφεύγουμε crashes, δίνουμε κατανοητά μηνύματα στον χρήστη.

Άσκηση 3.7 45 λεπτά

Γράψτε τουλάχιστον 6 tests για το Smart Notes (δημιουργία, αναζήτηση, διαγραφή, κενός τίτλος, διαγραφή ανύπαρκτης σημείωσης, αναζήτηση χωρίς αποτελέσματα). Τουλάχιστον 2 tests να τα γράψετε χωρίς AI.

Άσκηση 3.8 1 ώρα

Σας δίνεται έκδοση του Smart Notes με 4 bugs. Βρείτε και διορθώστε τα, γράφοντας για καθένα ένα test που αποτυγχάνει πριν τη διόρθωση και περνά μετά.

Άσκηση 3.9 30 λεπτά

Αποθηκεύστε τις σημειώσεις σε αρχείο JSON με System.Text.Json, ώστε να μη χάνονται όταν κλείνει η εφαρμογή. Ξεκινήστε από ένα σύντομο intent.md και γράψτε πρώτα το test.

MODULE 4 · 4 ΩΡΕΣ

Git, code review και ομαδική δουλειά

Ο συμμετέχων δουλεύει όπως σε πραγματική ομάδα: κάθε αλλαγή είναι ορατή, ελεγμένη και αναστρέψιμη.

4.1 **Version control με Git:** Commit, branch, merge, επιστροφή σε προηγούμενη έκδοση. Γιατί είναι το «δίχτυ ασφαλείας» όταν δουλεύουμε με AI.

4.2 **GitHub workflow:** Remote repository, pull request, conflicts. Continuous integration με GitHub Actions: dotnet build και dotnet test σε κάθε pull request.

4.3 **Code review:** Τι ελέγχει ο reviewer, πώς σχολιάζουμε, πώς δεχόμαστε σχόλια. Ο κώδικας του AI περνά από review όπως κάθε άλλος.

4.4 **AI ως reviewer:** AI review σε κάθε pull request ως πρώτο επίπεδο ελέγχου. Τι εντοπίζει καλά (style, προφανή bugs) και τι όχι (business λογική, αρχιτεκτονικές αποφάσεις). Την τελική έγκριση την κρατά άνθρωπος.

4.5 **Τεκμηρίωση ως γνώση της ομάδας:** README, XML documentation comments, commit messages και CLAUDE.md, που κρατά κανόνες και αποφάσεις της ομάδας σε μορφή που διαβάζει και το AI. Χρήση AI για πρώτο draft, με έλεγχο από εμάς.

4.6 **Ευθύνη:** Ο κώδικας που κάνουμε commit είναι δικός μας, όποιος κι αν τον έγραψε. Διαφάνεια στην ομάδα για τη χρήση AI.

Άσκηση 4.7 1 ώρα

Ανεβάστε το Smart Notes σε GitHub repository με README που εξηγεί εγκατάσταση και χρήση και με CLAUDE.md για το project. Στήστε GitHub Actions workflow που τρέχει dotnet build και dotnet test. Προσθέστε δυνατότητα επεξεργασίας σημείωσης σε ξεχωριστό branch και ανοίξτε pull request.

Άσκηση 4.8 30 λεπτά

Σε ζευγάρια, κάντε review στο pull request του άλλου. Αφήστε τουλάχιστον 3 σχόλια και απαντήστε σε όσα λάβατε.

MODULE 5 · 4 ΩΡΕΣ

Ασφάλεια, απόρρητο και κανονιστικό πλαίσιο

Ο συμμετέχων ξέρει τι δεν μπαίνει ποτέ σε ένα AI εργαλείο και πώς ελέγχει την ασφάλεια του κώδικα.

5.1	**Δεδομένα και απόρρητο:** Τι γίνεται με ό,τι επικολλάμε σε ένα εργαλείο. Προσωπικά δεδομένα, δεδομένα πελατών, κώδικας εταιρείας. Σύνδεση με GDPR.
5.2	**Secrets:** API keys και κωδικοί δεν μπαίνουν σε κώδικα ούτε σε prompts. User Secrets (dotnet user-secrets) στην ανάπτυξη, environment variables στην παραγωγή, appsettings.json χωρίς κωδικούς, .gitignore.
5.3	**Συνηθισμένα κενά ασφαλείας σε AI κώδικα:** SQL injection (και πώς το αποτρέπουν οι παραμετρικές queries και το Entity Framework Core), έλλειψη ελέγχου input, hardcoded credentials, ξεπερασμένες βιβλιοθήκες.
5.4	**Dependencies:** Ανύπαρκτα ή κακόβουλα NuGet packages που προτείνει το AI. Έλεγχος πριν το install, dotnet list package –vulnerable.
5.5	**Άδειες χρήσης:** Open source licenses, πνευματικά δικαιώματα σε κώδικα που παράγεται από AI.
5.6	**Κανονιστικό πλαίσιο:** EU AI Act σε γενικές γραμμές, εσωτερικές πολιτικές χρήσης AI, ιδιαίτερες απαιτήσεις σε ρυθμιζόμενους κλάδους όπως ο τραπεζικός.
5.7	**Governance την ώρα που δουλεύει το AI:** Permissions και hooks στα agentic εργαλεία ως approval gates (π.χ. μπλοκάρισμα commit που περιέχει secret, έγκριση πριν από εντολές που αλλάζουν αρχεία). Ανθρώπινο review για κρίσιμο και ρυθμιζόμενο κώδικα.

Άσκηση 5.8 45 λεπτά

Κάντε security review στο Smart Notes και σε ένα δοσμένο κομμάτι κώδικα C# με 5 κενά ασφαλείας. Για κάθε εύρημα: περιγραφή, κίνδυνος, διόρθωση.

Άσκηση 5.9 45 λεπτά

Γράψτε μια σελίδα «κανόνες χρήσης AI» για μια φανταστική ομάδα 5 developers: τι επιτρέπεται, τι απαγορεύεται, τι χρειάζεται έγκριση. Μετατρέψτε έναν κανόνα σε αυτόματο έλεγχο, ως hook στο agentic εργαλείο ή ως βήμα στο CI.

MODULE 6 · 6 ΩΡΕΣ

Χτίζοντας εφαρμογές με LLM APIs

Ο συμμετέχων περνά από χρήστη AI εργαλείων σε developer που ενσωματώνει AI σε δική του εφαρμογή.

6.1 **Από console app σε Web API:** HTTP requests, JSON, authentication με API key. ASP.NET Core Minimal APIs: το Smart Notes αποκτά endpoints για τις σημειώσεις. Το LLM ως υπηρεσία.

6.2 **Η πρώτη κλήση σε LLM API από C#:** .NET SDK του παρόχου ή Microsoft.Extensions.AI (IChatClient), messages, system prompt, παράμετροι όπως max tokens και temperature, dependency injection του client.

6.3 **Structured output:** Ζητάμε απάντηση σε JSON, την κάνουμε deserialize σε C# record με System.Text.Json και την ελέγχουμε πριν τη χρησιμοποιήσουμε.

6.4 **Error handling:** Timeouts, rate limits, retries με HttpClient resilience handlers, τι βλέπει ο χρήστης όταν η κλήση αποτύχει.

6.5 **Κόστος και απόδοση:** Χρέωση ανά token, επιλογή μοντέλου ανάλογα με την εργασία, caching.

6.6 **Prompt ως κώδικας:** Τα prompts της εφαρμογής μπαίνουν σε version control και ελέγχονται όπως ο υπόλοιπος κώδικας.

Άσκηση 6.7 — 1 ώρα

Προσθέστε στο Smart Notes API δύο endpoints: αυτόματη περίληψη μιας σημείωσης σε 2 προτάσεις και αυτόματη κατηγοριοποίηση (Δουλειά, Προσωπικό, Ιδέες) με απάντηση σε JSON. Υποχρεωτικά: το API key από User Secrets ή environment variable, error handling για αποτυχία κλήσης και μη έγκυρο JSON, και xUnit tests με mock IChatClient αντί για πραγματική κλήση.

Άσκηση 6.8 — 30 λεπτά

Τρέξτε την κατηγοριοποίηση σε 20 σημειώσεις με δύο διαφορετικά μοντέλα. Συγκρίνετε ακρίβεια, χρόνο απόκρισης και κόστος, και προτείνετε ποιο θα χρησιμοποιούσατε και γιατί.

MODULE 7 · 6 ΩΡΕΣ

RAG, tool use και agents

Ο συμμετέχων καταλαβαίνει τα τρία patterns που βρίσκονται πίσω από τις περισσότερες σημερινές AI εφαρμογές.

7.1 **Γιατί το LLM δεν ξέρει τα δικά μας δεδομένα:** Training cutoff, ιδιωτική πληροφορία, όριο context window.

7.2 **Retrieval-Augmented Generation (RAG):** Embeddings μέσω Microsoft.Extensions.AI ή του API του παρόχου, ομοιότητα κειμένων (cosine similarity σε C#), αναζήτηση των πιο σχετικών κομματιών, απάντηση με παραπομπή στην πηγή.

7.3 **Tool use / function calling:** Ορίζουμε C# μεθόδους ως tools, το μοντέλο ζητά να εκτελεστεί μία, ο κώδικάς μας την εκτελεί και επιστρέφει το αποτέλεσμα.

7.4 **Agents:** Επανάληψη σκέψη–ενέργεια–έλεγχος, πότε αξίζει και πότε προσθέτει μόνο πολυπλοκότητα. Agentic coding εργαλεία στην πράξη: plan mode, subagents που δουλεύουν παράλληλα σε ξεχωριστά κομμάτια, πόση αυτονομία τους δίνουμε.

7.5 **Σύνδεση με εξωτερικά συστήματα:** Η ιδέα πίσω από πρωτόκολλα όπως το Model Context Protocol (MCP) και το επίσημο C# SDK του.

7.6 **Human in the loop:** Ποιες ενέργειες χρειάζονται επιβεβαίωση από άνθρωπο πριν εκτελεστούν.

Άσκηση 7.7 — 1 ώρα

Προσθέστε στο Smart Notes API endpoint «Ρώτα τις σημειώσεις μου». Η εφαρμογή βρίσκει τις 3 πιο σχετικές σημειώσεις, απαντά με βάση μόνο αυτές και δείχνει ποιες χρησιμοποίησε. Αν δεν βρει σχετική σημείωση, το λέει αντί να μαντέψει.

Άσκηση 7.8 — 45 λεπτά

Προσθέστε ένα tool «δημιούργησε σημείωση», ώστε ο χρήστης να γράφει «θύμισέ μου να πάρω τηλέφωνο τον λογιστή την Πέμπτη» και η εφαρμογή να δημιουργεί τη σημείωση μετά από επιβεβαίωση.

MODULE 8 · 6 ΩΡΕΣ

Evaluation, παραγωγή και τελικό project

Ο συμμετέχων ελέγχει αν μια AI λειτουργία δουλεύει αξιόπιστα και παρουσιάζει την ολοκληρωμένη εφαρμογή του.

8.1 **Γιατί τα AI features θέλουν διαφορετικό έλεγχο:** Η απάντηση δεν είναι ίδια κάθε φορά, άρα ένα απλό assert δεν αρκεί.

8.2 **Evals:** Σύνολο ερωτήσεων με αναμενόμενες απαντήσεις, κριτήρια επιτυχίας, βαθμολόγηση από άνθρωπο ή από δεύτερο μοντέλο. Τα evals τρέχουν συνεχώς ως βήμα στο CI, όχι μόνο στο τέλος.

8.3 **Guardrails:** Έλεγχος input και output, όρια στο τι επιτρέπεται να κάνει η εφαρμογή.

8.4 **Monitoring και feedback:** Καταγραφή κλήσεων, κόστους και λαθών με ILogger και OpenTelemetry, feedback από χρήστες.

8.5 **Responsible AI:** Διαφάνεια προς τον χρήστη ότι μιλά με AI, bias, πότε μια απόφαση πρέπει να μείνει σε άνθρωπο.

8.6 **Κλείνοντας τον κύκλο:** Όταν μια μέτρηση βγει εκτός αποδεκτών ορίων (π.χ. πτώση ακρίβειας στα evals ή αύξηση κόστους), γράφουμε νέο intent.md και η αλλαγή ξαναπερνά από spec, tests και review. Ανακεφαλαίωση του AI-native SDLC: intent, spec, build, test, deploy, maintain.

8.7 **Η καριέρα του developer με AI:** Ποιες δεξιότητες μένουν κεντρικές (σχεδιασμός, έλεγχος, επικοινωνία), πώς συνεχίζουμε να μαθαίνουμε.

Άσκηση 8.8 1 ώρα

Φτιάξτε eval με 10 ερωτήσεις για τη λειτουργία «Ρώτα τις σημειώσεις μου» και προσθέστε το στο GitHub Actions, ώστε το build να αποτυγχάνει αν η ακρίβεια πέσει κάτω από ένα όριο που θα ορίσετε. Μετρήστε, αλλάξτε το prompt και μετρήστε ξανά.

Τελικό project 10 λεπτά παρουσίαση

Παρουσιάστε το Smart Notes: demo, αρχιτεκτονική, το intent.md και το spec μιας λειτουργίας, αποτελέσματα eval, ένα σημείο όπου το AI έκανε λάθος και πώς το εντοπίσατε, και τι θα αλλάζατε για να μπει σε παραγωγή.

Πρακτικές πληροφορίες

ΗΜΕΡΟΜΗΝΙΑ ΕΝΑΡΞΗΣ

25/1/2027

ΗΜΕΡΟΜΗΝΙΑ ΛΗΞΗΣ

10/3/2027

ΗΜΕΡΕΣ ΚΑΙ ΩΡΕΣ

Δευτέρα · Τετάρτη · Παρασκευή, 15:00–17:00

ΤΡΟΠΟΣ

Online μέσω MS Teams

Για την επιτυχή ολοκλήρωση απαιτείται η παρακολούθηση όλων των ενοτήτων, που θα διαρκέσουν 40 ώρες (20 συναντήσεις των 2 ωρών) και θα διεξαχθούν μέσω MS Teams. Το link συμμετοχής θα σας σταλεί με την εγγραφή σας. Δίνεται η δυνατότητα δικαιολογημένων απουσιών έως και 8 (οκτώ) διδακτικών ωρών.

Στο πλαίσιο της εκπαίδευσης θα διεξαχθούν ομαδικές ασκήσεις, θα δοθεί χρόνος για θεωρία και πρακτική και θα υλοποιηθούν tests που θα οδηγήσουν στην αξιολόγηση των υποψηφίων.

Βεβαίωση: με την επιτυχή ολοκλήρωση της εκπαίδευσης, θα λάβετε Βεβαίωση Παρακολούθησης της Εκπαιδευτικής Ακαδημίας Application Development powered by Natech Banking Solutions.

Natech BANKING SOLUTIONS

Natech Banking Solutions
www.natechbanking.com · info@natechbanking.com