WEITERBILDUNGSANTRAG

BASTA! Herbst 2026, Mainz

Business Case für die Teilnahme einer Entwickler:in

Zusammenfassung

Die BASTA! Herbst 2026 fällt in ein Zeitfenster, das für uns technisch entscheidend ist: **.NET 11.0 erscheint im November 2026** – die Konferenz liegt unmittelbar davor. Wir erfahren alles über SDK-Änderungen, Erweiterungen der Basisklassen sowie Auswirkungen auf WPF und Windows Forms, *bevor* wir migrieren, nicht wenn der Build bereits rot ist.

Darüber hinaus adressiert das Programm vier Themen mit direkter finanzieller Wirkung für uns: Sicherheits- und Haftungsrisiken bei KI-Integrationen, Kostenkontrolle bei Modell- und Function-Calling-Aufrufen, messbare Produktivitätsgewinne durch Agent-Workflows sowie übertragbare Architektur- und Governance-Muster. Die ganztägigen Workshops liefern ein einsetzbares Setup zum Mitnehmen, kein reines Konsumformat.

Angefragt wird: Teilnahme inklusive Workshop-Tage, Reise und Unterkunft. Im Gegenzug verpflichte ich mich zu einem internen Recap und einem konkreten Pilotprojekt.

1. Timing: .NET 11.0 erscheint im November 2026

Die Konferenz liegt direkt vor dem Release. Behandelt werden .NET 11.0, C# 15, Entity Framework Core 11, ASP.NET Core 11 und Blazor 11 – jeweils mit Fokus auf SDK-Erweiterungen, API-Änderungen und Migrationspfade.

- **Planungssicherheit:** Wir planen unsere Migration auf Basis erklärter Breaking Changes statt auf Basis nachträglich gelesener Release Notes.
- **Desktop-Risiko:** Für unsere Windows-Desktop-Anteile (WPF/WinForms) gibt es eine eigene Session. Das ist genau der Bereich, in dem Upgrades erfahrungsgemäß am teuersten scheitern.
- **Budgetwirkung:** Wir wissen vor der Budgetplanung 2027, welcher Migrationsaufwand realistisch ist.

2. Risiko & Compliance: die teuerste Lücke schließen

Vier Sessions adressieren direkt unsere finanzielle und rechtliche Exposure. Fehlende Autorisierung und unsichere KI-Integration führen laut Programm zu Datenverlust, DSGVO-Verstößen, Bußgeldern und Reputationsschäden.

Thema	Konkretes Risiko	Was die Session liefert
OWASP Top Ten für KI	Prompt Injection, Datenexfiltration aus Kontexten, unsichere Function Calls	Angriffsflächen und Schutzmaßnahmen für LLM-Einsatz in .NET-Backends und Web-APIs
Broken Access Control (BOLA, BOPLA, BFLA)	Weiterhin Risiko Nr. 1 bei APIs – besonders kritisch, wenn Agents auf Backend-APIs zugreifen	Autorisierungs-Patterns auf Objekt- und Property-Ebene mit Code-Beispielen
OWASP Top Ten 2025 für ASP.NET Core und Blazor	Neu priorisierte Risiken in unserem Haupt-Stack	Sichere Defaults, SAST-Pipelines, Runtime-Monitoring
MCP – die unsichtbare Gefahr	Unkontrolliertes Function Calling als Sicherheits- und Kostenvorfall	Kontrollmechanismen für Architekten: Allow-Lists, Context Redaction

Einordnung: Ein einziger vermeidbarer Sicherheitsvorfall übersteigt die Kosten der Teilnahme um ein Vielfaches. Diese Position ist damit weniger Weiterbildungsbudget als Risikovorsorge.

3. Kosten: unkontrollierte KI-Rechnungen vermeiden

Das Programm behandelt explizit das Szenario, das im Controlling für Überraschungen sorgt: ungebremste Function Calls und zu häufige Modellabfragen. Insbesondere bei multimodaler Nutzung oder großen Kontexten treiben die Cloud-Rechnungen nach oben. Ohne Raten- und Budgetkontrolle wird das bei jeder MCP-Integration akut.

Vermittelt werden konkrete Gegenmaßnahmen:

- Rate Limiter und Circuit Breaker für Function Calls
- Allow-Lists für zulässige Funktionen
- Context Redaction vor dem Modellaufruf
- Budgetierung und Kostenbegrenzung als Architekturmuster

Hinzu kommen die Nacharbeitskosten fehlerhaften KI-Codes: Gelangt unsicherer Agent-Code in Produktion, entstehen Rework-, Test- und Incident-Kosten. Die Session zur „Self-Defending Codebase“ zeigt eine CodeRule Engine, die Verstöße bereits beim Editieren erkennt und auch KI-generierten Code validiert.

4. Produktivität: übertragbare Workflows statt Buzzwords

Workflow	Wirkung auf unsere Lieferfähigkeit
Copilot Review Agents als PR-Gate	Automatisierte Reviews inklusive Security-Checks laufen vor dem menschlichen Reviewer. Nur komplexe oder risikobehaftete PRs gehen an Menschen – Review-Durchsatz und Release-Tempo steigen.
Azure DevOps + GitHub Copilot MCP	Work Items, Pull Requests, Builds und Releases reichern den Agent-Kontext an. Kontextreichere Agents liefern brauchbarere und sicherere Vorschläge.
Specification Driven Development	Aufwand verlagert sich in die Spezifikationsphase, Agents erzeugen den Implementierungscode. Weniger Kontext-Switching, weniger Wiederholarbeit.
LLM-gestützte SAST und Quick-Fix-Generatoren	Bessere Fehlerentdeckung und kürzere Fix-Loops in der Testphase, niedrigere Time-to-Remediate.
CodeRule Engine / Self-Defending Codebase	Inline-Policy-Checks für Architektur, APIs und Coding-Standards. Automatisiert zusätzlich das Onboarding neuer Entwickler.

Ehrliche Einordnung: Die Sessions benennen auch die Grenzen –„Human in the Loop“ bleibt notwendig, MCP-Server und Agent-Konfiguration erfordern Initialaufwand und Agents produzieren gelegentlich fehlerhaften Code. Genau deshalb ist begleitete Einführung günstiger als Trial-and-Error im laufenden Projekt.

5. Architektur & Unabhängigkeit

- **Cloud-native .NET-Apps ohne Vendor Lock-in (Radius):** Deployment-Strategie, die uns nicht an einen Anbieter bindet.
- **Authentication for Micro Frontends:** BFF-Pattern für verteilte Frontends.
- **Routenbasierte Architektur und Ownership:** Klare Zuständigkeiten in wachsenden Codebases.
- **Angular 2026:** Signals, Signal Forms, zone-less-Betrieb und Testing mit Vitest im Browser-Mode.
- **Web-Standards:** WebAssembly-Fortschritte und Built-in AI im Browser, aus Sicht des W3C-Architekturboards – relevant für unsere SPA-Architekturentscheidungen.

Quellen

Alle genannten Sessions stammen aus dem Programm der BASTA! Herbst 2026 (Mainz):

- Neuigkeiten in .NET 11.0
- Angular in 2026 – Was ist neu? Was ist anders?
- Was ist neu im Web? 2026 Edition
- Sichere Apps mit ASP.NET Core und Blazor: Die OWASP Top Ten 2025
- Sichere Intelligenz: Die OWASP Top Ten(s) für KI
- Broken Access Control: BOLA, BOPLA und BFLA
- MCP – Die unsichtbare Gefahr in der KI-Integration
- GitHub Copilot: Die Agents sind da
- KI mit Azure DevOps nutzen: GitHub Copilot + Azure DevOps MCP
- Specification Driven Development: Wie KI-Agenten Softwareentwicklung neu definieren
- Codequalität vs. KI: Chancen, Grenzen und Erfahrungen in der Qualitätssicherung
- The Self-Defending Codebase: CodeRule Engine
- Cloud-native .NET-Apps deployen – ohne Vendor Lock-in mit Radius
- Authentication for Micro Frontends, Done Right
- Route-basierte Architektur, Struktur und Ownership in wachsenden Frontend-Anwendungen
- Workshop: Smarter Coden mit KI – Modernes AI Coding Setup für Entwickler
- Workshop: Webanwendung mit Blazor 10.0/11.0 – Tipps, Tricks und Tuning
- Workshop: Programmierer und Architekten – Strategien 2026

Programmübersicht: basta.net/mainz/programm-mainz/