# Trivue Platform - Arhitektura Sistema

## 📊 Vizuelni Dijagrami

Kompletni vizuelni dijagrami arhitekture su dostupni na sledećoj lokaciji: **→ Detaljni vizuelni dijagrami arhitekture**

Dijagrami pokrivaju:

- High-Level System Architecture sa svim modulima
- Triple-Layer Event-Driven Architecture flow
- ICrossModuleEvent Pattern implementaciju
- Hexagonal Architecture deep dive
- gRPC Internal API arhitekturu
- Module Communication preko event-a
- Database Schema-per-Module pristup
- Microservices Migration Path
- Event Flow Decision Matrix

## Uvod

Trivue predstavlja modernu loyalty platformu koja transformiše način na koji kompanije grade i održavaju odnose sa svojim korisnicima. Nakon godina rada sa raznim loyalty rešenjima, shvatili smo da je potreban sistem koji kombinuje fleksibilnost, skalabilnost i jednostavnost održavanja. Trivue nije samo još jedna loyalty platforma - to je pažljivo dizajniran ekosistem koji omogućava kompanijama da grade dublje veze sa korisnicima kroz personalizovane nagrade, poene i kampanje.

Platforma je razvijena sa jasnom vizijom: kreirati sistem koji može da počne kao monolitna aplikacija za brz razvoj i deployment, ali koji je arhitekturalno spreman za transformaciju u mikroservise kada biznis potrebe to zahtevaju. Ova dvojnost nije slučajnost - rezultat je iskustva sa projektima koji su ili počeli previše jednostavno pa kasnije patili od tehničkog duga, ili počeli previše kompleksno pa nikad nisu ugledali produkciju.

## Arhitekturna Filozofija

Trivue koristi **Hybrid Event-Driven Modular Monolith** arhitekturu - pristup koji kombinuje najbolje iz tri sveta: jednostavnost modular monolitha, fleksibilnost mikroservisa i pouzdanost event-driven sistema. Ovo nije kompromis, već strateška odluka koja omogućava evoluciju sistema u skladu sa biznis potrebama.

### Domain-Driven Design (DDD) u Srcu Sistema

Sistem je organizovan oko 12 jasno definisanih biznis domena (bounded contexts), gde svaki modul predstavlja koherentnu biznis celinu. Members modul upravlja korisnicima i njihovim profilima, Points modul vodi računa o poenima i transakcijama, Rewards modul definiše nagrade i njihovu razmenu, dok Integration modul omogućava komunikaciju sa eksternim sistemima poput POS terminala.

Svaki modul striktno sledi DDD principe:

- **Agregati** enkapsuliraju biznis logiku i garantuju konzistentnost
- **Value objekti** modeluju koncepte bez identiteta (Email, MemberId, PointsAmount)
- **Domain eventi** omogućavaju komunikaciju između agregata i modula
- **Repository pattern** apstrahuje perzistenciju

- **Result pattern** elegantno hendluje biznis greške bez bacanja exception-a

Ovaj pristup garantuje da biznis logika ostaje čista i nezavisna od infrastrukturnih detalja. Na primer, Member agregat zna kako da registruje novog člana, dodeli mu karticu i pokrene odgovarajuće domain evente, ali ne zna ništa o bazi podataka ili API endpoint-ima. (ADR-006)

### Hexagonal Architecture za Čistu Separaciju

Svaki modul sledi hexagonal (ports and adapters) arhitekturu sa četiri jasno definisana sloja:

**Domain Layer** - srce modula koje sadrži čistu biznis logiku bez ikakvih zavisnosti. Ovde žive agregati, value objekti, domain eventi i repository interfejsi. Domain nikad ne zavisi od Infrastructure sloja - to je nepovredivo pravilo.

**Application Layer** - orkestrira biznis operacije kroz CQRS pattern. Commands menjaju stanje sistema kroz agregate, Queries čitaju podatke optimizovane za prikaz, a Event Handlers reaguju na domain evente. Svaki handler koristi MediatR za asinhrono procesiranje sa CancellationToken podrškom. (ADR-002)

**Infrastructure Layer** - implementira tehničke detalje poput EF Core repository-ja, RabbitMQ consumer-a i Redis cache servisa. Ovaj sloj zavisi od Domain sloja (implementira njegove interfejse) ali Domain nikad ne zavisi od njega - perfektna inverzija zavisnosti.

**Presentation Layer** - eksponuje funkcionalnosti kroz REST API kontrolere ili gRPC servise. Kontroleri su tanki - samo primaju request, prosleđuju ga Application sloju kroz MediatR i vraćaju response.

### CQRS Pattern za Optimalne Performanse

Sistem striktno razdvaja write (Command) i read (Query) operacije. Commands prolaze kroz domain agregate garantujući biznis invarijante, dok Queries direktno čitaju denormalizovane podatke optimizovane za prikaz. Ovaj pristup omogućava različite optimizacije - Commands koriste transakcije i domain logiku, dok Queries koriste Redis cache i projekcije. (ADR-002)

```
// Command menja stanje kroz agregat
var member = await _memberRepository.GetByIdAsync(command.MemberId);
member.AssignCard(command.CardNumber);
await _memberRepository.UpdateAsync(member);

// Query čita direktno iz denormalizovanih podataka
var memberProfile = await _dbContext.MemberProfiles
    .Where(m => m.Id == query.MemberId)
    .ProjectTo<MemberProfileDto>(_mapper.ConfigurationProvider)
    .FirstOrDefaultAsync();
```

## Triple-Layer Event-Driven Architecture

Platforma koristi sofisticirani trolajernu event-driven arhitekturu koja kombinuje brzinu, pouzdanost i performanse. Svaki sloj ima specifičnu ulogu i karakteristike. (ADR-008)

### Layer 1: MediatR (Instant Feedback)

Prvi sloj omogućava instant komunikaciju između modula kroz in-memory MediatR publish/subscribe mehanizam. Kada korisnik promeni ime lokacije, svi povezani POS uređaji odmah vide promenu - ispod 1ms latencija. Backoffice administratori ne moraju da refreshuju stranicu - promene su vidljive odmah.

Ovaj sloj koristi se za operacije gde je brzina kritična - denormalizacija podataka, cache invalidacija, real-time notifikacije. Međutim, MediatR handleri su dizajnirani da budu "throwaway" - ako handler failuje, ne blokira glavnu operaciju već loguje grešku i pušta Layer 2 da sinhronizuje podatke kroz eventual consistency model.

### Layer 2: RabbitMQ (Guaranteed Delivery)

Drugi sloj garantuje isporuku event-a kroz RabbitMQ message broker. Hangfire background job svakih 30 sekundi procesira Outbox tabelu i šalje neprocesuirane evente na RabbitMQ. Svaki modul ima svoj consumer koji sluša relevantne evente kroz Inbox Pattern.

Ključna karakteristika ovog sloja je idempotentnost - consumer prvo proverava da li je podatak već sinhronizovan (možda kroz Layer 1) pre nego što izvrši update. Ovo omogućava self-healing - ako Layer 1 failuje, Layer 2 će eventualno sinhronizovati podatke. Retry logika sa eksponencijalnim backoff-om garantuje da će event biti procesiran čak i ako je servis privremeno nedostupan.

### Layer 3: Redis Cache (Performance Optimization)

Treći sloj ubrzava čitanje kroz Redis distributed cache. Kad god se podatak promeni, MediatR handler (Layer 1) odmah invalidira relevantne cache ključeve. Sledeći GET request će imati cache miss i učitati fresh podatke iz baze koji se potom kešuju za buduće čitanje.

Cache strategija je granularna - umesto invalidiranja celog cache-a, targetiraju se samo relevantni ključevi. Na primer, promena imena lokacije invalidira `location:{id}` i `pos-devices:location:{id}` ali ne utiče na ostale keširane podatke.

### ICrossModuleEvent Pattern

Ključna inovacija u našoj arhitekturi je ICrossModuleEvent marker interface koji elegantno rešava problem duplog procesiranja cross-module event-a. Eventi označeni ovim interfejsom se automatski preskaču u Layer 1 (MediatR) i procesiraju se isključivo kroz Layer 2 (RabbitMQ). (ADR-020)

```
// Cross-module event se procesira samo jednom
public sealed record LocationUpdatedDomainEvent(
    Guid LocationId,
    string LocationName) : DomainEvent, ICrossModuleEvent;
```

Ovo rešenje eliminisalo je database lock contention, ubrzalo API response vremena i pojednostavilo debugging. Sistem sada jasno razlikuje lokalne evente (procesiraju se instant kroz MediatR) i cross-module evente (procesiraju se asinhrono kroz RabbitMQ sa eventual consistency).

## gRPC Internal API - Opcioni Četvrti Sloj

Za specifične use case-ove gde je potrebna sinhorna komunikacija sa garantovanim odgovorom, platforma koristi gRPC kao opcioni četvrti sloj. Notifications modul koristi gRPC da dobije member podatke pre slanja emaila, Campaigns modul validira target member-e, a POS integracija dobija real-time podatke sa minimalnom latencijom. (ADR-021)

gRPC server koristi iste MediatR query handler-e kao REST API, što garantuje konzistentnost biznis logike. Client implementacija uključuje automatski fallback na HTTP API ako gRPC nije dostupan. Ovaj pristup omogućava 10-20x brže read operacije (2-5ms vs 50-100ms) za kritične sinhrone potrebe.

```
// gRPC sa automatskim fallback na HTTP
public async Task<MemberDto?> GetMemberAsync(Guid memberId)
{
    if (_featureFlags.UseGrpcForMembers)
    {
        try
        {
            return await _grpcClient.GetMemberAsync(memberId);
        }
        catch (RpcException)
        {
            // Fallback na HTTP
        }
    }
    return await _httpClient.GetFromJsonAsync<MemberDto>
($"/api/members/{memberId}");
}
```

Važno je napomenuti da gRPC služi isključivo za READ operacije - nikad ne menja stanje sistema. Port 5020 je rezervisan za interni gRPC saobraćaj i nikad se ne eksponuje van mreže.

## Moduli i Njihove Odgovornosti

Platforma se sastoji od 12 pažljivo dizajniranih modula, gde svaki ima jasnu biznis odgovornost:

### Members Module (Referentna Implementacija)

Upravlja korisnicima, njihovim profilima i segmentacijom. Ovaj modul predstavlja zlatni standard za DDD implementaciju sa čistom separacijom domain logike od infrastrukture. Member agregat enkapsulira sve biznis invarijante vezane za registraciju, verifikaciju i upravljanje članstvom.

### Points Module

Vodi računa o poenima, transakcijama i earning rule-ovima. Implementira kompleksne algoritme za kalkulaciju poena baziranih na različitim kriterijumima (iznos kupovine, kategorija proizvoda, vreme transakcije). Svaka transakcija je immutable - jednom kreirana, ne može se menjati, što garantuje audit trail.

### Products Module

Katalog proizvoda sa kategorijama, atributima i pricing strategijama. Podržava hijerarhijske kategorije, varijante proizvoda i kompleksne pricing rule-ove. Integriše se sa POS sistemima za sinhronizaciju proizvoda.

### Rewards Module

Definiše nagrade, uslove za otkključavanje i mehanizme razmene. Podržava različite tipove nagrada (popusti, proizvodi, iskustva) sa fleksibilnim pravilima kvalifikacije. Achievement sistem motiviše korisnike kroz gamifikaciju.

### Referrals Module

Upravlja referral programima i tracking-om preporuka. Automatski generiše jedinstvene referral kodove, prati konverzije i dodeljuje nagrade i referrer-u i referee-u. Fraud detection mehanizmi sprečavaju zloupotrebu.

### Locations Module

Upravlja fizičkim lokacijama, radnim vremenom i geo podacima. Svaka promena lokacije automatski se propagira na sve povezane POS uređaje kroz event-driven mehanizam. Podržava multi-store scenario sa centralizovanim upravljanjem.

### Vouchers Module

Generisanje, validacija i redempcija vaučera. Svaki vaučer ima jedinstveni kod sa kriptografski sigurnom generacijom. Podrška za različite tipove (single-use, multi-use, time-limited) sa real-time validacijom.

### Campaigns Module

Orkestrator za marketing kampanje sa targetiranjem i praćenjem. Omogućava kreiranje kompleksnih kampanja sa više koraka, A/B testiranje i real-time analytics. Integriše se sa email i SMS provider-ima.

### Analytics Module

Real-time i batch analytics sa dashboard-ima. Koristi CQRS read model za brze upite nad denormalizovanim podacima. Event streaming omogućava real-time metrike bez uticaja na transakcione performanse.

### Integration Module (Inbox Pattern Referenca)

Centralni hub za komunikaciju sa eksternim sistemima. Implementira Inbox Pattern za reliable event processing sa idempotent handler-ima. HMAC autentifikacija garantuje sigurnu komunikaciju sa POS terminalima.

### Inventory Module

Upravljanje fizičkim karticama sa serijskim brojevima. Batch import omogućava učitavanje hiljada kartica odjednom. Lifecycle management prati karticu od kreiranja do aktivacije i eventualnog isteka.

### Notifications Module

Orkestracija email, SMS i push notifikacija. Template engine omogućava personalizovane poruke. Retry mehanizmi garantuju isporuku. Preference management poštuje korisničke želje.

## Tehnički Stack

Platforma koristi moderne, production-ready tehnologije:

### Backend (.NET 9)

- **ASP.NET Core** za REST API sa Minimal API pristupom gde je moguće
- **Entity Framework Core** sa Code-First migracijama i shadow properties
- **MediatR** za CQRS pattern i in-process event handling
- **FluentValidation** za deklarativnu validaciju sa reusable rule-ovima
- **AutoMapper** za mapiranje između slojeva sa projection optimizacijama
- **Hangfire** za background job-ove sa persistent queue
- **RabbitMQ** za message broker sa automatic retry i dead letter queue
- **Redis** za distributed caching i session storage
- **PostgreSQL 17** kao primarna baza sa JSONB podrškom za fleksibilnost
- **Serilog** strukturirano logovanje sa Seq sink-om
- **gRPC** za internu komunikaciju između modula

### Frontend (Next.js 16)

- **React 19** sa Server Components za optimalne performanse
- **TypeScript** za type safety kroz ceo frontend
- **Tailwind CSS** za brz i konzistentan UI development
- **React Query** za server state management sa optimistic updates
- **React Hook Form** sa Zod validacijom
- **Recharts** za data vizualizaciju
- **Radix UI** headless komponente za accessibility

### Infrastructure

- **Docker** kontejnerizacija sa multi-stage build-ovima
- **Kubernetes** orkestracija sa horizontal pod autoscaling
- **GitHub Actions** CI/CD pipeline sa automatskim testiranjem
- **Azure Application Insights** za monitoring i telemetriju
- **Seq** centralizovano logovanje sa strukturiranim upitima
- **NGINX** reverse proxy sa SSL termination

## Metrički Pregled

Platforma predstavlja ozbiljan inženjerski poduhvat sa impresivnim metrikama:

- **12 biznis modula** sa jasnim bounded context-ima
- **44 .NET projekta** organizovanih po clean architecture principima
- **1,508 C# fajlova** u modularnoj strukturi
- **~122,412 linija koda** sa visokim kvalitetom i pokrivenošću testovima
- **28 arhitekturnih odluka (ADR)** dokumentovanih i implementiranih
- **1,437 unit i integration testova** sa 100% pass rate
- **71 React komponenta** sa TypeScript definicijama
- **25+ custom hook-ova** za reusable frontend logiku

## Mikroservisna Budućnost

Arhitektura platforme nije slučajno dizajnirana - svaki aspekt je promišljen sa ciljem lake transformacije u mikroservise kada biznis potrebe to zahtevaju.

### Zašto je Transformacija Laka?

**Moduli su već nezavisni** - svaki modul ima sopstvenu bazu (schema), svoj DbContext, svoje repository-je. Ne dele entitete, ne pozivaju se direktno, komuniciraju samo preko event-a. U suštini, već se ponašaju kao mikroservisi samo što su deploy-ovani zajedno.

**Event-driven komunikacija je spremna** - RabbitMQ je već u upotrebi za Layer 2. Samo treba podesiti queue-ove i exchange-ove za cross-service komunikaciju umesto cross-module. Outbox i Inbox pattern-i već garantuju reliable messaging.

**Bounded context-i su jasni** - DDD pristup garantuje da svaki modul ima jasne granice. Nema shared kernel-a (osim osnovnih value object-a koje svaki servis može duplicirati). Agregati enkapsuliraju biznis logiku koja ostaje netaknuta.

### Šta je Potrebno za Transformaciju?

**API Gateway** će biti entry point za sve servise. Kong ili Ocelot mogu da ruutuju request-e, hendluju autentifikaciju i rate limiting. Gateway također omogućava verzionisanje API-ja i canary deployment.

**Service Discovery** omogućava servisima da se dinamički pronalaze. Consul ili Kubernetes native service discovery eliminišu hard-coded URL-ove. Health check-ovi automatski uklanjaju nezdrave instance.

**Distributed Tracing** postaje kritičan za debugging. OpenTelemetry sa Jaeger-om omogućava praćenje request-a kroz sve servise. Correlation ID-jevi povezuju logove između servisa.

**Circuit Breakers** štite sistem od kaskadnih failure-a. Polly library omogućava retry logiku, timeout-e i fallback mehanizme. Bulkhead pattern izoluje failure-e.

**Configuration Management** centralizuje konfiguraciju. Azure App Configuration ili Consul KV omogućavaju dinamičke promene bez redeployment-a. Feature flag-ovi kontrolišu postupnu migraciju.

### Benefiti Mikroservisne Arhitekture

**Nezavisno skaliranje** - Points modul može da ima 10 instanci tokom Black Friday-ja dok Members modul ima samo 2. Auto-scaling bazirano na CPU/memory/queue length optimizuje troškove.

**Nezavisni deployment** - bugfix u Rewards modulu ne zahteva deployment cele platforme. Blue-green deployment eliminiše downtime. Rollback je instant ako nešto pođe po zlu.

**Tehnološka raznolikost** - Analytics modul može da koristi Python i Pandas za data science. Integration modul može da koristi Go za performanse. Svaki tim bira najbolji alat za svoj domen.

**Fault isolation** - greška u Campaigns modulu ne ruši celu platformu. Circuit breaker izoluje problematičan servis. Graceful degradation održava core funkcionalnosti.

**Tim autonomija** - svaki tim potpuno poseduje svoj servis od razvoja do produkcije. Brže donošenje odluka, manje koordinacije, veća produktivnost.

### Plan Migracije (6-12 meseci)

**Faza 1 (Mesec 1-2):** Postavljanje infrastrukture - Kubernetes cluster, API Gateway, Service Discovery, Distributed Tracing. Sve ostaje monolith ali kroz gateway.

**Faza 2 (Mesec 3-4):** Ekstraktovanje Integration modula kao prvi mikroservis. Već komunicira samo preko event-a, ima jasnu odgovornost. Pilot za testiranje infrastrukture.

**Faza 3 (Mesec 5-8):** Postepeno ekstraktovanje modula po prioritetu. Analytics i Notifications su dobri kandidati - stateless, jasne granice. Points i Members ostaju za kraj zbog kompleksnosti.

**Faza 4 (Mesec 9-10):** Optimizacija komunikacije. Uvođenje gRPC za sinhrone pozive. Event streaming sa Kafka za high-throughput. Cache layer sa Redis za cross-service data.

**Faza 5 (Mesec 11-12):** Monitoring i optimizacija. Performance tuning, cost optimization, disaster recovery. Chaos engineering za testiranje resilience.

## Sigurnosni Aspekti

Bezbednost nije afterthought već integralni deo arhitekture:

**HMAC autentifikacija** za POS integraciju garantuje da samo autorizovani uređaji mogu da komuniciraju sa platformom. Svaki request je potpisan sa secret key-em, timestamp-om sprečava replay napade.

**JWT tokeni** sa refresh token mehanizmom za korisnike. Access token-i su short-lived (15 minuta), refresh token-i se čuvaju sigurno i mogu se opozvati. Role-based i claim-based autorizacija.

**Encryption at rest** za sensitive podatke. PII (Personally Identifiable Information) se enkriptuje u bazi. Ključevi se rotiraju periodično. GDPR compliance ugrađen od početka.

**Rate limiting** sprečava abuse. Različiti limiti za različite endpoint-e. Distributed rate limiting preko Redis-a. Graceful degradation kada se dostigne limit.

**Audit logging** prati sve kritične operacije. Ko je šta uradio i kada. Immutable log-ovi za compliance. Integration sa SIEM sistemima.

## Performance Optimizacije

Platforma je optimizovana za high-throughput i low-latency:

**Database optimizacije** - Indeksi na svim foreign key-evima i često korišćenim kolonama. Composite indeksi za složene upite. Partial indeksi za velike tabele. Regular VACUUM i ANALYZE.

**Caching strategija** - Multi-layer caching (in-memory, Redis, CDN). Cache-aside pattern sa automatic invalidation. Sliding expiration za često korišćene podatke.

**Async sve** - Svi I/O pozivi su async sa ConfigureAwait(false). Paralelno procesiranje gde je moguće. Bulk operacije za batch scenario.

**Connection pooling** - za bazu i Redis. Optimalni pool size baziran na load testing-u. Connection reuse smanjuje overhead.

**Query optimizacije** - Projection za čitanje samo potrebnih kolona. Include za eager loading i izbegavanje N+1 problema. Raw SQL za kompleksne upite gde EF Core nije efikasan.

## Testing Strategija

Kvalitet je garantovan kroz comprehensive testing:

**Unit testovi** pokrivaju svu biznis logiku u Domain sloju. Aggregate invarijante, value object validacije, domain servisi. Mocking za eksterne zavisnosti.

**Integration testovi** verifikuju komunikaciju između slojeva. WebApplicationFactory za API testove. TestContainers za realnu bazu i RabbitMQ. Transakcioni rollback za čistu test izolaciju.

**End-to-end testovi** simuliraju realne korisničke scenarije. Playwright za browser automatizaciju. API chain testing za složene workflow-e.

**Performance testovi** sa k6 i NBomber. Load testing identifikuje bottleneck-e. Stress testing određuje breaking point. Soak testing verifikuje memory leak-ove.

**Contract testovi** garantuju kompatibilnost između modula. Pact za consumer-driven contracts. Schema validacija za event-e.

## Monitoring i Observability

Platforma je dizajnirana za produkciju sa comprehensive monitoring-om:

**Application Insights** prati performance, failure rate, dependency calls. Custom metrike za biznis KPI-jeve. Alerting na anomalije.

**Structured logging** sa Serilog omogućava brzu dijagnostiku. Correlation ID povezuje logove kroz ceo request. Log aggregation u Seq-u sa naprednim query-ima.

**Health checks** za sve komponente (baza, Redis, RabbitMQ). Liveness i readiness probe za Kubernetes. Graceful shutdown čeka da se završe in-flight request-i.

**Dashboard-ovi** u Grafani vizualizuju metrike. Business dashboard-i prate KPI-jeve. Technical dashboard-i prate infrastrukturu.

**Distributed tracing** sa Application Insights prati request kroz sve slojeve. Dependency map vizualizuje komunikaciju. Performance profiler identifikuje spore metode.

## DevOps i CI/CD

Automatizacija omogućava brz i siguran deployment:

**Git flow** sa feature, develop i main branch-evima. Pull request-ovi zahtevaju code review. Automated checks (build, test, lint) pre merge-a.

**CI pipeline** pokreće se na svaki commit. Build, unit testovi, integration testovi, code analysis. Docker image se build-uje i push-uje u registry.

**CD pipeline** deploy-uje na environment-e. Dev deployment na svaki merge u develop. Staging deployment sa smoke testovima. Production deployment sa approval gate-om.

**Infrastructure as Code** sa Terraform-om. Sva infrastruktura je verzionisana. Moduli za reusable komponente. State se čuva u cloud storage.

**Secret management** sa Azure Key Vault. Aplikacija nikad ne vidi raw secret-e. Automatic rotation za kritične secret-e. Audit trail za sve access-e.

## Zaključak

Trivue platforma predstavlja moderan pristup razvoju enterprise software-a. Kombinacija modular monolith arhitekture sa event-driven pristupom omogućava brz razvoj i deployment dok istovremeno priprema sistem za buduću evoluciju ka mikroservisima.

Trolajernu event-driven arhitektura sa ICrossModuleEvent pattern-om elegantno rešava klasične probleme distributed sistema - garantuje isporuku poruka, omogućava instant feedback korisniku i optimizuje performanse kroz caching. Opcioni gRPC sloj dodaje fleksibilnost za sinhrone operacije gde su neophodne.

Sa 12 dobro definisanih modula, 1,508 C# fajlova organizovanih po clean architecture principima i preko 1,400 testova, platforma je spremna za produkciju. Monitoring, logging i comprehensive DevOps pipeline garantuju smooth operacije.

Najvažnije, arhitektura nije kruta - dizajnirana je za evoluciju. Kada biznis potrebe prerastu trenutni modular monolith, transformacija u mikroservise će biti prirodan korak, ne kompletan redizajn. Moduli su već nezavisni, komunikacija je event-driven, bounded context-i su jasni.

Trivue nije samo tehnološko dostignuće - to je biznis enabler koji omogućava kompanijama da grade meaningful odnose sa korisnicima kroz personalizovane loyalty programe. Arhitektura garantuje da tehnologija neće biti bottleneck za biznis inovacije.

---

**Zoran Stevovic, Senior Software Architect** *Datum: 2025-11-13*