



Систем електронског плаћања заснован на микросервисној архитектури

Electronic payment system based on microservice architecture

Дарие Чолак, Факултет техничких наука, Нови Сад

Студијски програм – РАЧУНАРСТВО И АУТОМАТИКА

Кратак садржај – У овом раду је описано пројектовање, имплементација и примена система за електронско плаћање заснованог на модерној микросервисној архитектури у облаку. Имплементацијом овог систем посебан акценат се ставља на обраду трансакција путем више метода плаћања, укључујући плаћање банковном картицом, криптовалутама, PayPal налогом и QR кодом. Иницијатор трансакција у овом систему електронског плаћања је онлајн веб продавница. За потребе продукције систем је мигриран на Amazon Web Services (AWS) платформу.

Кључне речи: систем електронског плаћања, микросервиси, трансакција, AWS

Abstract – This paper describes the design, implementation, and deployment of an electronic payment system based on a modern cloud-based microservices architecture. Through the implementation of this system, particular emphasis is placed on processing transactions via multiple payment methods, including credit card payments, cryptocurrencies, PayPal accounts, and QR codes. The initiator of transactions in this electronic payment system is an online web store. The system was migrated to the Amazon Web Services (AWS) platform for production.

Keywords: electronic payment system, microservices, transaction, AWS

НАПОМЕНА: Овај рад проистекао је из мастер рада чији ментор је био др Срђан Вукмировић, ред. проф.

1. УВОД

Системи електронских плаћања представљају једно од најдинамичнијих подручја савремене ИТ индустрије. Глобални раст е-трговине, мобилних апликација и дигиталних финансијских услуга створио је потражњу за системима који морају бити високо доступни, сигурни, скалабилни и способни да симултано обрађују трансакције, од класичних банковних картица до савремених криптовалута. Традиционални монолитни приступи таквих система показао се незадовољавајуће када се суочавају са захтевима

модерног тржишта, највише у погледу еластичности, независног скалирања компоненти и брзог увођења нових функционалности.

Посебна пажња током развоја посвећена је фази продукције, седам независних микросервиса мигрирани су на AWS платформу. Сваки од микросервиса поседује своју сопствену базу података, поред тога кориснички (frontend) интерфејс поседују три микросервиса.

2. АРХИТЕКТУРА СИСТЕМА

Имплементирани систем електронског плаћања је пројектован као дистрибуирани систем са јасно развојеним слојевима одговорности. На највишем нивоу, његова архитектура се састоји од три концептуална слоја: корисничког слој (frontend), позадинског слој (backend) и слој за перзистенцију података. Кориснички слој имплементиран је у React окружењу. За имплементацију backend микросервиса коришћен је ASP.NET Core, док Microsoft SQL Server континуирано бележи податке.

Расподела микросервиса:

- *Telecom webshop* – веб продавница оператера за телекомуникације који иницира плаћање, комуницира са остатком система путем API Gateway-a.
- *API Gateway* - централна улазна тачка за све захтеве, имплементира рутирање.
- *Payment Service Provider (PSP)* - кординише ток плаћања и баласирање система (load balancing).
- *Bank сервис* - комбинација корисничког (frontend) и позадинског слој (backend) за банкарски портал.
- *Payment Card Center (PCC)* - посреднички сервис (middleware) који омогућава трансакције између две различите банке,
- *PayPal сервис* – служи за интегрисање са PayPal платформом.
- *Coin сервис* – служи за интегрисање са CoinPayments платформом.

Сваки позадински (backend) сервис поштује принцип једне базе по сервису (database per service), што значи да ниједан сервис не приступа директно бази података другог сервиса. Све информације које један

сервис треба од другог прибављају се искључиво кроз дефинисане *API* позиве. Овај приступ, познат као лабава спрега (*loose coupling*), осигурава да промена шеме базе података једног сервиса не захтева измене на другим сервисима.

2.1. Теоријска основа

Микросервисна архитектура (*microservices architecture*) је приступ развоју софтвера у којем се апликација структурира као скуп малих, независно распоређених сервиса, од којих сваки имплементира специфичну функционалност, комуницира са другим сервисима путем добро дефинисаних *API*-ја и управља сопственим подацима. Систематизацијом карактеристика микросервисне архитектуре, истичу се следеће кључне особине [1]:

- Организација око пословних способности - сваки сервис енкапсулира једну пословну област (банковно плаћање, *PayPal* интеграција, плаћање криптовалutom).
- Децентрализовано управљање подацима - сваки сервис поседује и контролише своју базу података.
- Независно постављање - сервис се може ажурирати, скалирати и рестартовати без утицаја на остале сервисе.
- Отпорност на отказ (*Fault Tolerance*) - систем мора бити способан да настави рад када један или више сервиса постане недоступно.

У савременом контексту, архитектура система за обраду картичних плаћања темељи се на концепту четворостране шеме (*four-party scheme*), у којој учествују: купац (власник картице), продавац, банка продавца (*acquirer bank*) и банка купца (*issuer bank*) [2].

2.2. Технологије

У имплементацији система електроског плаћања су корићене следеће технологије:

- *ASP.NET Core* - вишеплатформских (*cross-platform*) *framework* отвореног кода (*open-source*), намењен развоју модерних *cloud*-оријентисаних веб апликација, развијен од стране *Microsoft*-а.
- *React* - *JavaScript* библиотека отвореног кода (*open-source*) за изградњу корисничких интерфејса, коју је развио *Meta (Facebook)* [3].
- *Microsoft SQL Server* - релациони систем за управљање базама података (*RDBMS*) компаније *Microsoft* [4].
- *Entity Framework Core (EF Core)* - објектно-релациони мапер (*object-relational mapper*) за приступ бази података.
- *HashiCorp Consul* – механизам за динамичку регистрацију и проналажење сервиса у мрежи.
- *Ocelot* - *.NET* библиотека отвореног кода (*open-source*) која имплементира образац *API Gateway* за потребе микросервисне архитектуре [5].

- *Docker* - платформа за контејнеризацију апликација [6].

2.3. Ток плаћања

Централни пословни процес система електронског плаћања јесте ток обраде плаћања. Постоје четири основе варијанте овог тока, у живности која метода плаћања је избарана. Оно сто је заједничко свим методама је иницијални корак који се одвија у *PSP*-у. Плаћање картицом банке одвија се у следећим корацима, прво купац у *Telecom webshop*-у бира жељени пакет и иницира плаћање. Друго *Telecom webshop* шаље захтев *PSP* сервису путем *API Gateway*-а, наводећи износ и валуту трансакције. Треће, *PSP* генерише јединствени идентификатор трансакције, купац бира начин плаћања картицом банке и *PSP* прослеђује купца на страницу за унос података картице. Четврто, купац уноси податке картице (број, име власника картице, датум истека, CVV код). Пето, врши се прелиминарна валидација података картице и прослеђује захтев за ауторизацијом *PCC* сервису. Шесто, *PCC* сервис идентификује банку издаваоца на основу *BIN* кода картице и прослеђује захтев одговарајућем банковном сервису (*BankOne* или *Bank Two*). Седмо, *Bank* сервис проверава стање рачуна, доступна средства и врши дедукцију износа трансакције. Осмо, резултат ауторизације (одобрено/одбијено, ауторизациони код) враћа се истим путем. Девето, *PSP* бележи резултат трансакције и обавештава *Telecom webshop* о исходу. На крају, купац се преусмерава на страницу која представља листу пакета на које се претплатио. Овај ток плаћања важи у случају када банка купца и банка продавца се разликују. У случају када су банке исте, долази до интерног преноса новца и тада се *PCC* сервис не користи. Поред класичног плаћања картицом, имплементирано је плаћање *QR* кодом. Купац може у својој мобилној апликацији банке да отвори плаћање *IPS QR* кодом и скенира *QR*. Када купац одабере *PayPal* као методу плаћања, *PSP* преусмерава захтев ка *PayPal* сервису, који конструише захтев према *PayPal REST API*-ју. *PayPal* враћа *URL* за ауторизацију. Након што купац ауторизује плаћање у свом *PayPal* налогу, *PayPal* шаље повратни позив (*callback*) на *PayPal* сервис, затим даље иде на *PSP*, који потврђује трансакцију и евидентира резултат. За плаћања криптовалутама, *Coin* сервис генерише јединствену *crypto* адресу за сваку трансакцију. Купцу се приказује *QR* код са адресом и износом у одређеној криптовалuti. Сервис прати *blockchain* за долазне трансакције на генерисаној адреси и након потврде (*confirmation*) обавештава *PSP* о успешности плаћања.

3. МИКРОСЕРВИСИ

У имплементираном систему електронског плаћања јасно су раздвојене одговорности сваког микросервиса, сваки микросервис покрива тачно одређену пословну функционалност и независно се развија, тестира и поставља. У наставку су детаљно описани сви микросервиси који чине систем, њихове

одговорности и начин на који се уклапају у целокупни платни ток.

3.1. Telecom сервис

Telecom сервис покрива три области одговорности. Прва најважнија област је управљањем претплатама и инцијализација плаћања према *PSP* сервису. Друга је управљањем корисницима, корисници се региструју и пријављују, а при успешној пријави добијају *JWT* токен који корсите за приступ заштићеним ресурсима. Трећа област је управљање листом пакета, оператор дефинише понуду телеком пакета. Ток плаћања на *Telecom* сервису одвија се у две фазе. Када купац одабере пакет и покрене куповину са *frontend*-а, претплата се одмах креира у бази са привременим статусом, а затим се иницира трансакција према *PSP* - у који преузима контролу над током плаћања. Након што *PSP* заврши обраду и пошаље повратну информацију о исходу, *Telecom* сервис ажурира статус претплате, активира је у случају успешног плаћања или је означава као неуспешну.

3.2. PSP сервис

PSP сервис је централна координациона компонента читавог система, једини посредник између трговца (*Telecom webshop*-а) и свих доступних платних сервиса. Свака трансакција у систему, без обзира на методу плаћања, обавезно пролази кроз *PSP*. Када *Telecom* иницира плаћање, *PSP* валидира продавца, креира трансакцију са јединственим идентификатором и преусмерава купца на *PSP frontend* за одабир методе. Након одабира, *PSP* проналази одговарајући *plugin*, мења статус трансакције у "обрада" и делегира даље обраду том *plugin*-у. Трансакција пролази кроз дефинисани животни циклус, од чекања, преко обраде, до једног од коначних стања, успешно завршена, неуспешна и отказана. Након што платни сервис заврши обраду, шаље повратни позив *PSP* -у са резултатом. *PSP* проналази трансакцију, ажурира њен статус и уколико је плаћање успешно, асинхронно обавештава *Telecom* да активира претплату.

3.3. Bank сервис

Bank сервис имплементира пословну логику банке и покреће се у две независне инстанце, *BankOne* и *BankTwo*, поседује и свој *frontend*. Свака од банака користи засебну *SQL Server* базу података. Инстанца која се покреће контролисана је од стране *environment* варијабле, што значи да се исти бинарни код базе покреће за обе банке, а разлика је искључиво у конфигурацији. Овај приступ демонстрира како се хоризонтално скалирање једног сервиса може остварити чистом конфигурацијом параметара без икаквог дуплирања кода. У оквиру *Bank* сервиса имплементирано је *QR* код плаћање.

3.4. PCC сервис

Ток обраде међубанкарског плаћања одвија се у неколико корака. *Bank* сервис банке продавца шаље захтев *PCC* сервису када утврди да картица не припада њеном *BIN* опсегу. *PCC* евидентира трансакцију, проналази банку купца из базе по *BIN* коду и прослеђује комплетан захтев купчевој банци. Банка

купца проверава стање рачуна, одобрава или одбија трансакцију и враћа одговор са сопственим идентификатором поруџбине и са временском ознаком. *PCC* тај одговор обрађује и шаље назад банци продавца.

3.5. Coin сервис

Coin сервис је микросервис задужен за интеграцију система са платном мрежом базираном на криптовалутама. Сервис користи *CoinPayments* платформу као посредника за обраду крипто трансакција и библиотеке *NBitcoin* и *QRCoder* за генерисање *coin* адреса и *QR* кодова. Сервис имплементира детерминистички систем генерисања *coin* адреса заснован на *HD (Hierarchical Deterministic) wallet* принципу.

3.6. PayPal сервис

За једнократна плаћања, сервис креира поруџбину као заузету (*capture*), што значи да се средства резервишу тек у тренутку када корисник одобри плаћање, а не пре тога. Одговор *PayPal* сервера садржи листу *link*-ова, од којих сервис извлачи *approve* линк и враћа га клијенту. Корисник бива преусмерен на *PayPal* страницу за одобравање, а након што потврди плаћање, врши се заузеће поруџбине, *PayPal* тада терети корисников рачун и трансакција се сматра завршеном.

4. AWS ИНФРАСТРУКТУРА

Постовка система електронског плаћања у фазу продукције реализован је на *Amazon Web* сервису. Комбинујући *EC2* инстанце за микросервисе и *RDS* за управљање базама података, систем се репрезентује у *cloud* инфраструктури. Прелазак са локалног окружења на *cloud* инфраструктуру захтевало је низ прилагођавања конфигурација, али није захтевало промене у изворну коду сервиса. Једини сервис који је наставио сам локалним извршавањем је *Telecom*.

4.1. AWS EC2

Amazon Elastic Compute Cloud (Amazon EC2) је централни рачунарски сервис платформе *Amazon Web*, који омогућава изнајмљивање виртуелних сервера у облаку [7]. Тип *EC2* инстанце која је искоришћена за потребе овог пројекта је *t3.micro* и она се покреће на *Ubuntu AMI*. Регион на којем су подизане *EC2* инстанце је Франкфурт. За потребе продукције коришћена је стабилна *IP* адреса, статичке *IPv4* адресе које се трајно резервишу на *AWS* налогу и могу се у било ком тренутку прикључити или одвојити. Две *EC2* инстанце унутар исте *VPC* мреже комуницирају директно путем *IP* адресе кроз *AWS* мрежу.

4.2. AWS RDS

Amazon Relational Database Service (Amazon RDS) је сервис релационих база података у *cloud*-у који аутоматизује административне задатке као што су интегрисање софтвера базе података, примена *patch*-ева, прављење резервних копија, праћење рада и опоравак од грешака [8]. Пиликом креирања *RDS* инстанце, *AWS* нуди аутоматско конетковање са одговарајућом *EC2* инстанцом. За потребе овог

система електронског плаћања одабран је *RDS Microsoft SQL Server*. За продукциона окружења *RDS* нуди *Multi-AZ* конфигурацију у којој се аутоматски креира *standby* реплика у другој зони доступности унутар исте регије. Будући да овај систем електронског плаћања у локалном контејнеру користи *Entity Framework Code-First* приступ, процес подизања локалне контејнерске базе на *RDS* се своди креирање празне *RDS* инстанце. Поред тога потребно је променити конекциони *string* у *environment* варијаблама *Docker* контејнера, након чега сервис при првом покретању аутоматски примењује све миграције на празну *RDS* инстанцу и креира комплетну шему базе података.

5. МЕРЕЊЕ ЛАТЕНЦИЈЕ

За мерење *end-to-end* латенције сваког платног тока коришћен је *.NET Stopwatch* механизам који прати један захтев кроз све микросервисице од *Telecom* сервиса до финалног одговора. Свака метода плаћања мерена је кроз 10 узастопних мерења у контролисаним условима, а приказане вредности представљају просек добијених резултата. Важно је напоменути да измерене вредности обухватају целокупан ток, од тренутка иницијализације плаћања до пријема финалног статуса, укључујући време корисничке интеракције са интерфејсима.

Табела 1. Резултати мерења

Метода плаћања	Време (s)
QR плаћање	50
PayPal плаћање	70
Картично плаћање	270
Coin плаћање	420

5.1. Анализа резултата

Време *QR* плаћања приказано у табели 1, представља најбржи метод плаћања, генерише се *QR* код према *NBS IPS* стандарду, корисник скенира код помоћу мобилне апликације и потврђује трансакцију. Измерено време је очекивано уз претпоставку да је корисник спреман са отвореном апликацијом. *PayPal* време од 70 секунди је реална за корисника који је већ аутентификован у *PayPal* сесији у *browser*-у, у случају потребе за пријавом време би било знатно дуже. Картично плаћање показује време ручног уноса комплетних података картице, број картице (16 цифара), име власника, датум истека и *CVV* код. Плаћање криптовалутама је очекивано најспорији ток, време од око 420 секунди потребно је за потврду трансакције на *LTCT* тестној мрежи и како би систем добио нотификацију од стране *CoinPayments* платформе.

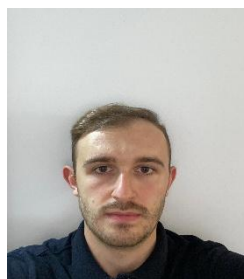
6. ЗАКЉУЧАК

Примена микросервисне архитектуре показала се као права одлука у домену система електронског плаћања. Сваки сервис могао се тестирати и развијати независно, отказ једног сервиса није утицао на остале делове система. Пребацивање локалних *Docker* слика кроз *Docker Hub* на *AWS* платформу покзао се као ефикасан приступ и осигурало је конзистентност података. Кроз израду овог система, показало се да за развој финансијских апликација, права је одлука одабир микросервисне архитектуре у комбинацији са *AWS* платформом. Даљи развој система могао би бити увођење *message broker*-а као што су *Apache Kafka* или *RabbitMQ* који би побољшао отпорност система на краткотрајне отказе, због чувања порука у реду (*queue*). Увођење система за праћење *Grafana* пружио би бољи увид у перформансе система у фази продукције.

7. ЛИТЕРАТУРА

- [1] S. Newman, *Building Microservices: Designing Fine-Grained Systems*, 2nd ed. Sebastopol, CA, USA: O'Reilly Media, 2021.
- [2] D. S. Evans and R. Schmalensee, *Paying with plastic: the digital revolution in buying and borrowing*. MIT Press, 2004.
- [3] A. Banks and E. Porcello, *Learning React: Modern Patterns for Developing React Apps*. Sebastopol, CA, USA: O'Reilly Media, 2020.
- [4] <https://learn.microsoft.com/en-us/sql/sql-server/what-is-sql-server> (pristupljeno u junu 2026.)
- [5] C. de la Torre, *Reference Application: .NET Microservices: Architecture for Containerized .NET Applications*, 4th ed. Redmond, WA, USA: Microsoft Corporation, 2020.
- [6] <https://docs.docker.com/get-started/docker-overview> (pristupljeno u junu 2026.)
- [7] <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html> (pristupljeno u junu 2026.)
- [8] <https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html> (pristupljeno u junu 2026.)

Кратка биографија:



Дарие Чолак рођен је у Панчеву 1998. године. Уписао је Факултет техничких наука 2017. године, смер Примењено софтверско инжењерство. Уписао је мастер студије на студијском програму Рачунарство и аутоматика.
Контакт:
colakdarie@gmail.com