Зборник радова Факултета техничких наука, Нови Сад

УДК: 004.9
ДОИ: https://doi.org/10.24867/02BE41Todic

MICROSOFT AZURE-ОВА АРХИТЕКТУРА БЕЗ СЕРВЕРА

MICROSOFT AZURE SERVERLESS ARCHITECTURE

Десанка Тодић, *Факултет техничких наука, Нови Сад*

Област – РАЧУНАРСТВО И АУТОМАТИКА

Кратак садржај – *У раду су описане архитектуре система информационих технологија од првих система грађених само традиционалном монолитном архитектуром, затим преко система грађених помоћу микросервиса до систем без сервера и коришћења готове инфраструктуре за градњу жељеног система. Акценат рада је стављен на архитектуру без сервера* ***Microsoft Azure*** *и његове* ***Azure*** *функције које су детаљно објашњене.*

Кључне ријечи: *традиционална монолитна архитектура, микросервисна архитектура, архитектура без сервер, cloud computing,* ***Azure*** *функције, платформа као сервис,* ***Azure*** *портал,* ***API***

Abstract – *The paper describes the architecture of the information technology system from the first systems built only by traditional monolithic architecture, then through the systems built using microservice to the serverless architecture and using the ready infrastructure to build the desired system. The focus of work is on the Microsoft Azure serverless architecture and its Azure Functions that are explained in detail.*

Keywords: *traditional monolithic architecture, microservice architecture, serverless architecture, cloud computing, Azure functions, platform as a service, Azure portal, API*

1. УВОД

Од алата преко машина до рачунара, људи траже начине за аутоматизовање поновљеног рада и стандардизацију задатака којима се том приликом баве тако да могу да се усредсреде на специјализоване доприносе завршавању задатака и рјешавању проблема. Паралелно с тим, индустрија информационих технологија је еволуирала, и људи су се трудили да повећају ефикасност на свим нивоима система, од централне процесорске јединице до фарми сервера, у нади да ће се остварити највећи излазни учинак система.

Архитектура без сервера је тачка у коју ова два тока конвергирају. То је тачка у којој је појединачни напор најсјајније фокусиран на специфичан задатак, а неупотребљивост система је сведена на минимум.

У свету без сервера, програмери стварају рјешења умјесто инфраструктуре и надгледају извршење. Не закупљује се најчешће беспослена виртуелне машина него се плаћају само временски интервали у којима она ради. У ***Microsoft Azure***-у постоји много потрошних сервиса које се могу повезати заједно како би се формирало комплетно рјешење. Кључна компонента архитектуре без сервера ***Microsoft Azure*** јесу ***Azure*** функције.

2. ТРАДИЦИОНАЛНА АРХИТЕКТУРА АПЛИКАЦИЈА

Већина традиционалних апликација изграђена је коришћењем трослојног шаблона архитектуре апликација: слој презентације (кориснички интерфејс), средњег слоја (функционална логика процеса) и слоја података (рачунарски подаци и приступ подацима). Сваки ниво ради на намјенском серверу и статички је конфигурисан са именом хоста (енг. ***hostname***) и ***IP*** (енг. ***Internet Protocol***) адресама сервера осталих нивоа од којих зависи. Ове апликације су распоређене на статичкој инфраструктури која се не мијења. Традиционалне апликације имају веома мало знања о инфраструктури на којој су покренуте и претпостављају да се инфраструктура неће промјенити или пропасти. Трослојна архитектура састоји се из сљедећих слојева:

- **Презентациони слој.** Ово је највиши ниво апликације. Ниво презентације приказује информације везане за такве услуге као што су прегледање робе, куповина и садржај корпе. Комуницира са другим нивоима помоћу којих се резултати преносе на слој претраживача/клијента и свих осталих нивоа у мрежи. Једноставно, то је слој који корисници могу директно приступати (као што је веб страница или графички кориснички интерфејс оперативног система).

- **Слој апликације (пословна логика, логички ниво или средњи ниво).** Логички ниво се извлачи из нивоа презентације и, као свој слој, контролише функционалност апликације примјеном детаљне обраде.
- **Слој података.** Ниво података укључује механизме перзистености података (сервери базе података, дијељење датотека итд.) и слој за приступ подацима који енкапсулира механизме перзистености и излаже податке. Ниво за приступ подацима треба да обезбиједи ***API*** (енг. ***Application programming interface***) нивоу

НАПОМЕНА:
Овај рад проистекао је из мастер рада чији ментор је био др Александар Купусинац, ванр. проф.

апликације да излаже методе за управљања складиштеним подацима без излагања или стварања зависности од механизама за складиштење података. Избјегавање зависности од механизама за складиштење дозвољава ажурирања или промјене без да то утиче на клијенате апликацијског слоја или да они буду упознати са промјеном. Као и код раздвајања било којег нивоа, постоје трошкови за имплементацију и често побољшана скалабилност и одрживост буде на рачуна опадања преформанси.

3. МИКРОСЕРВИСНА АРХИТЕКТУРА

*„Микросервис је техника развоја софтвера – варијанта архитектонског стила сервисно-орјентисане архитектуре (**SOA**[1]-а) који структурира апликацију као скуп услуга који се лако повезују“* [1]. У архитектури микросервиса, сервиси су заокружене цјелине намјењене обављању неке функционалности и протоколи којим се комуницира са овим сервисима имају само неколико правила и пракси, понекад и само једно које је лако пратити. Комуникација између микросервиса се обавља програмским интерфејсима ***API***-јима који су независни од програмског језика нпр. ***Representational State Transfer*** (***REST***) – архитектонски стил који дефинише скуп ограничења који се користе за креирање веб сервиса.

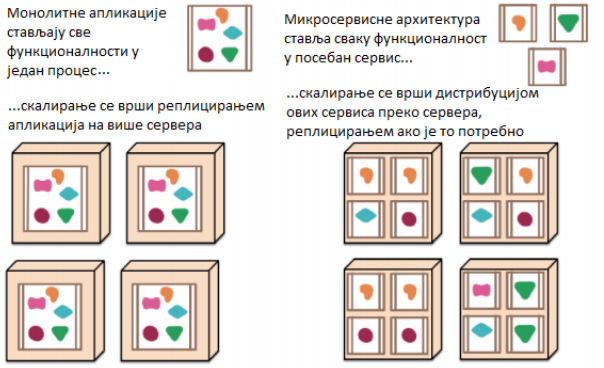
Најважнија предност разлагања апликације у различите мање сервисе је у томе што побољшава модуларност и чини апликацију лакшом за разумјевање, развој, тестирање и отпорност на разрушавање архитектуре. Такође, паралелним развојем омогућено је да мали аутономни тимова могу самостално да развијају своје сервисе. Због тога архитектуре базиране на микросервисима омогућавају непрекидну испоруку и примјену у систем.

Додавање нових функционалности у вишеслојној архитектури обично повећа сложеност бекенд (енг. ***backend***) слоја (пословна логика) и значајно компликује средњи слој. Код је ипак организован по модулима, па повећање броја функција доводи до повећања броја модула у средњем слоју апликације. Да би се одговорило на повећане захтјеве, скалирање се врши тако што се покреће више инстанци бекенд апликација са идентичним модулима.

Битне разлике:

- Нема гломазних модула са “шпагети” кодом.
- Компоненте се могу развијати потпуно независно једна од друге.
- Различити програмски језици, извршна окружења, оперативни системи, хардвер, база података.
- Компоненте се могу релативно једноставно мијењати.
- Може се замијенити и комплетна технологија модула.
- Могуће различито скалирање различитих компоненти.

Разлике у скалирању између традиционалне (монолитне) и микросервисне архитектуре представљене и објашњене су на слици 1.



Слика 1. Разлике у скалирању

4. АРХИТЕКТУРА БЕЗ СЕРВЕРА

У основи, архитектура без сервера представља следећу еволуцију сервиса рачунарства у облацима, изграђених на врху ***PaaS***-а (платформа као сервис), апстрахујући виртуалне машине, окружење апликације и екстерне зависности путем везивања тако да се програмери могу једноставно фокусирати на код за имплементацију пословне логике.

„Рачунарство у облацима је парадигма информационих технологија која омогућава свеобухватан приступ заједничким базама конфигурисаних системских ресурса и сервиса вишег нивоа који се брзо могу обезбиједити уз минималне напоре управљања, често преко интернета“ [3]. Три главне категорије сервиса рачунарства у облацима су: инфраструктура као сервиса (***IaaS***), платформа као сервис (***PaaS***) и софтвер као сервис (***SaaS***). *„Инфраструктура као сервиса (енг. **Infrastructure as a service**) је облик рачунарства у облацима који пружа виртуелизоване рачунарске ресурсе преко интернета“* [4].

*„Платформа као сервис (енг. **Platform as a service**) је комплетно окружење за развој и успостављање окружења у облацима, уз ресурсе који омогућавају да пруже све од једноставних апликација заснованих на облацима до софистицираних апликација које користе облаке“* [5].

Поред горе наведених основних облика рачунарства у облацима, важно је споменути и ***FaaS*** – функција као сервис (енг. ***Function as a service***) који представља еволуцију PaaS-а и на његовим принципима изграђена је архитектура без сервера са ***Azure*** функцијама.

*„Софтвер као сервис (енг. **Software as a service**) је модел дистрибуције софтвера у ком независни добављач (енг. **third-party provider**) хостује апликације и чини их доступним корисницима преко интернета”* [6].

[1] енг. ***Service-oriented architecture*** - архитектонски стил развоја софтвера гдје се сервиси који комуницирају међусобно пружају другим софтверским компонентама на услугу захваљујући комуникационом протоколу преко мреже.

4.1. ПРЕДНОСТИ АРХИТЕКТУРЕ БЕЗ СЕРВЕРА

Из пословне перспективе:

- Трошак који произлази из апликације без сервера заснован је на броју извршења функција, мјерено у милисекундама умјесто сатима.
- Омогућава агилност процеса тако што мање јединице које се могу распоређивати резултирају бржом испоруком функција на тржиште, повећавајући способност прилагођавања промјенама.
- Опадају трошкови запошљавања бекенд инжењера.
- Смањени оперативни трошкови пошто су сервери на клауду.

Из перспективе програмера:

- Смањена обавеза пошто нема бекенд инфраструктуре за коју би се могло бити одговорно.
- Лакше оперативно управљање.
- Брже постављање окружења.
- Скалабилно, нема потребе да се брине о броју истовремених захтјева.
- Подстиче иновације пошто је релативно нова архитектура.

Из перспективе корисника апликације:

- Ако компаније користе као конкурентску предност да брже испоручују нове функционалности апликације, онда корисници добијају нове функционалности брже него раније.
- Корисници могу лакше да обезбиједе сопствени бекенд за складиштење (нпр. ***Dropbox, Google Drive***).
- Апликације ове врсте могу да понуде кеширање на клијентској страни и да тако омогуће боље корисничко искуство када је ван мреже (енг. ***offline***).

4.2. МАНЕ АРХИТЕКТУРЕ БЕЗ СЕРВЕРА

Из пословне перспективе:

- Смањена укупна контрола пошто бекенд слој контролише продавац сервиса рачунарства у облацима.
- Може доћи до гашења сервиса који архитектура нуди.
- Сигурносни ризик пошто приступ свему може да има и клауд продавац.
- Сви ови недостаци могу бити ублажени алтернативама отвореног кода (енг. ***open-source***), али то повећава трошкове апликације.

Из перспективе програмера:

- Незрела технологија доводи до фрагментације компоненти и нејасних најбољих пракси.
- Архитектонска сложеност за функције које су превише сложене.
- Технички могуће да сусједне функције могу да преносе системске ресурсе испод хаубе, па да се појаве невалидни подаци.
- Тестирање локално је незгодно. Понекад је потребно изложити своју машину мрежи пребацивањем приватне у јавну адресу да би клауд сервис могао да комуницира са програмеровом машином.
- Трајање извршења је ограничено. Обично 300 секунди сервер има времена да изврши функцију пре него што саопшти да није успио у извршавању. Тако да функције не смију бити превише сложене.
- Недостатак оперативних алата, поготово за дибаговање.

Из перспективе корисника апликације:

- Ако се исправно не имплементира архитектура, апликација би могла пружити лоше корисничко искуство као резултат повећане учесталости захтјева.

5. АРХИТЕКТУРА БЕЗ СЕРВЕРА НА *AZURE* ПЛАТФОРМИ

Архитектура без сервера на ***Azure*** платформи укључује ***Azure*** функције, ***Logic Apps*** (апликације за логику) и ***Event Grid*** (мрежа догађаја). Ови сервиси раде заједно и повезују се са бројним другима ресурсима. Серверлес платформа функционише са свиме из базе и складишта до аналитике и машинског учења тј. вјештачке интелигенције.

Поред ових сервиса, могуће је користити и ***Application Insights*** (увид у апликацију), серверлес платформу за снимање дијагностичких трагова и телеметрије (технологија која омогућава даљинско мјерење и пренос информација од удаљених локација до оператера). ***Application Insights*** су доступни за апликације свих врста (десктоп, мобилне или веб) као и имплементације серверлеса.

6. АРХИТЕКТУРА БЕЗ СЕРВЕРА СА *AZURE* ФУНКЦИЈАМА

Архитектура без сервера омогућава чисту раздвојеност између кода и окружења у коме је хост. Имплементира се код у функцији која је позвана преко окидача. Након што се функција изврши, сви ресурси који су којој били потребни могу бити ослобођени. Окидач може да буде ручни, да се окида послије одређеног времена, ***HTTP*** захтјева, или отпремање датотеке. Резултат окидача је извршење кода. Иако сервери без сервера варирају, већина пружа приступ унапред дефинисаним ***API***-има и везама за поједностављивање задатака као што је додавање у базу података или резултате чекања.

Архитектура без сервера је архитектура која се у великој мјери ослања на апстракцији удаљеног окружења хоста да би се фокусирала на код. Може се сматрати као архитектура са мање сервера.

Рјешења заснована на контејнерима омогућава програмерима да користе постојеће скрипте за билд (енг. ***build*** – направити, подизати) да би обавили објаву кода до спремне слике серверлеса. Остале имплементације користе постојећа ***PaaS*** рјешења за пружање скалабилне архитектуре.

Апстракција значи да ***DevOps*** тим не мора да објезбеђује нити управља серверима нити специфичним контејнерима.

Постоје двије верзије ***Azure Functions*** фрејмворка. Стара верзија подржава комплетан ***.NET*** фрејмворк, а нова омогућава ***.NET Core*** и развој између платформи. Језици који су подржани поред ***C#*** су ***JavaScript***, ***F#*** и ***Java***, а ту је и експериментална подршка за ***Python***, ***PHP***, ***TypeScript***, ***Batch***, ***Bash*** и ***PowerShell***. Функције креиране на порталу објезбеђују синтаксу из скрипти. Функције креиране као самостални пројекти могу бити отпремљење на сервер уз потпуну подршку и могућности које пружа платформа.

7. ЗАКЉУЧАК

Овом раду описане су архитектуре помоћу којих се могу градити системи информационих технологија. Прва архитектура која је објашњена јесте традиционална монолитна архитектура. Затим слиједи описивање миркосервисне архитектуре које представља напредак у односу на традиционалну. Даље је објашњена архитектура без сервера са акцентом на архитектуру без сервера помоћу ***Azure*** функција. Ниједна од ових архитектура није савршена и има и предности и мане, стога у зависности од проблема који се ријешава, нека је боља, а нека лошија у односу на другу. За апликације које треба да ријешавају сложене проблеме, коначна архитектура обично обухвата више различитих архитектура и тиме се ствара хибридна апликација.

8. ЛИТЕРАТУРА

[1] Chen, Lianping (2018). Microservices: Architecting for Continuous Delivery and DevOps, *https://www.researchgate.net/publication/323944215_Microservices_Architecting_for_Continuous_Delivery_and_DevOps*

[2] Gautambiztalkblog, Microservices approach vs traditional approach, 2016, *https://gautambiztalkblog.com/tag/microservices-approach-vs-traditional-approach/*

[3] Wikipedia, Cloud computing, 2018, *https://en.wikipedia.org/wiki/Cloud_computing*

[4] Margaret Rouse, Infrastructure as a Service (IaaS), 2018, *https://searchcloudcomputing.techtarget.com/definition/Infrastructure-as-a-Service-IaaS*

[5] Microsoft Azure, What is PAAS, 2018, *https://azure.microsoft.com/en-us/overview/what-is-paas/*

[6] Margaret Rouse, Software as a Service (SaaS), 2016, *https://searchcloudcomputing.techtarget.com/definition/Software-as-a-Service*

[7] The agile admin, What is DevOps, 2017, *https://theagileadmin.com/what-is-devops/*

[8] Margaret Rouse, Internet of Things (IoT), 2018, *https://internetofthingsagenda.techtarget.com/definition/Internet-of-Things-IoT*

[9] Vangie Beal, Big Data, 2018, *https://www.webopedia.com/TERM/B/big_data.html*

[10] Robin Weston, Serverless architectures and continuous delivery, 2017, *https://www.gocd.org/2017/06/26/serverless-architecture-continuous-delivery/*

Кратка биографија:

Десанка Тодић рођена је 1994. године у Брчком, Босна и Херцеговина. Завршила је гимназију „Васо Пелагић“ у Брчком 2013. године. Факултет техничких наука у Новом Саду уписала је 2013. године. Испунила је све обавезе и положила све испите предвиђене студијским програмом.