Zbornik radova Fakulteta tehničkih nauka, Novi Sad

UDK: 4.9
DOI: https://doi.org/10.24867/31BE19Savic

MIGRACIJA MONOLITNE ARHITEKTURE NA BEZSERVERSKO OKRUŽENJE UZ KORIŠĆENJE AZURE SERVISA

MIGRATION FROM MONOLITE ARCHITECTURE TO A SERVERLESS PLATFORM USING AZURE SERVICES

Mihajlo Savić, *Fakultet tehničkih nauka, Novi Sad*

Oblast – ELEKTROTEHNIKA I RAČUNARSTVO

Kratak sadržaj – *U ovom radu opisana je migracija monolitne arhitekture na bezserversku infrastrukturu u cloud-u. U cilju omogućavanja efikasnog monitoringa, smanjenja troškova i povećane skalabilnost, implementirani su različiti Azure servisi, uključujući SQL Server, SQL Database, Function App.*

Ključne reči: *Monolitne, Cloud, Azure[7],*

Abstract – *In this paper, the migration from a monolithic architecture to a serverless infrastructure in the cloud is described. To enable efficient monitoring, reduce costs, and increase scalability, various Azure services have been implemented, including SQL Server, SQL Database, and Function App.*

Keywords: *Monolithic, Cloud, Azure[7]*

1. UVOD

Povećanjem infrastrukture i količine zahteva, odnosno potrebe za skalabilnošću i bržim odgovorima potrebna su nova rešenja, od kojih je trenutno najbolje bezserverska arhitektura. Ona omogućava korisniku kompletan fokus na poslovnu logiku, dok upravljanje resursima i infrastrukturom preuzima izabrani provajder.

U rad je uključeno istraživanje i pronalaženje najboljeg načina za migraciju postojeće monolitne arhitekture u bezserversku arhitekturu. Aplikacija predstavlja softver za podršku rada cvećare, kako bi se olakšalo kreiranje porudžbina i narudžbina, kao i praćenje rada zaposlenih.

Implementacija je omogućena korišćenjem Azure servisa, sa najvećim fokusom na Azure Function App, odnosno funkcije koje se nalaze i izvršavaju na cloud-u. Funkcije su pisane u .NET[2] okruženju, u C#[1] programskom jeziku i koriste SQL bazu podataka, koja se takođe nalazi na cloud-u.

Usled korišćenja studentske licence za Azure, pojedini servisi nisu implementirani (Azure Active Directory B2C, Azure Application Gateway...) te oni predstavljaju moguća unapređenja na plaćenim verzijama.

NAPOMENA:
Ovaj rad proistekao je iz master rada čiji mentor je bio dr Miroslav Zarić, vanr.prof..2. KORIŠĆENE

2. TEHNOLOGIJE

U ovom poglavlju opisani su ključni servisi koji su korišćeni za uspešnu implementaciju aplikacije u bezserverskom okruženju, kao i tehnologije korišćene u izradi gotove aplikacije.

2.1. KORIŠĆENE TEHNOLOGIJE

2.1.1. Angular[4]

Angular[4] je popularni open-source front-end framework koji se koristi za izgradnju dinamičnih web aplikacija. Razvijen je od strane Google-a i omogućava programerima da kreiraju jednostrane aplikacije (SPA - Single Page Applications) koje pružaju brže i interaktivnije korisničko iskustvo.

2.1.2. .NET[2]

.NET[2] je sveobuhvatan razvojni okvir (framework) koji omogućava izradu raznovrsnih aplikacija, uključujući web, desktop, mobilne i cloud aplikacije. Razvijen je od strane Microsoft-a i podržava više programskih jezika, najčešće C#[1], F# i VB.NET[2]. U ovom slučaju korišćen je C#[1] programski jezik.

2.1.3. Azure SQL Database[5]

Azure SQL Database je upravljana usluga baze podataka u cloudu koju nudi Microsoft Azure. To je relacijska baza podataka koja se oslanja na SQL Server tehnologiju, omogućavajući korisnicima da lako kreiraju, upravljaju i skaliraju baze podataka bez potrebe za održavanjem fizičke infrastrukture.

2.1.4. Azure Function App[8]

Azure Function App je serverless usluga koju nudi Microsoft Azure, koja omogućava razvoj i pokretanje malih, nezavisnih delova koda poznatih kao "funkcije". Ove funkcije se pokreću kao odgovor na različite događaje, omogućavajući programerima da reše specifične zadatke bez potrebe za upravljanjem infrastrukturom.

2.2. MOGUĆA UNAPREĐENJA

Usled korišćenja studentske licence neki od servisa nisu implementirani, te će oni biti navedeni kao moguća unapređenja na plaćenim verzijama.

2.2.1. Azure Active Directory B2C

Azure Active Directory B2C (Business to Consumer) je identitetska i pristupna usluga koju nudi Microsoft Azure, dizajnirana posebno za omogućavanje registrovanih korisnika da se autentifikuju i pristupaju web i mobilnim aplikacijama. Ova usluga omogućava preduzećima da upravljaju korisničkim identitetima i pristupom na jednostavan i siguran način.

2.2.2. Azure Application Gateway

Ova usluga je namenjena upravljanju saobraćajem za web aplikacije. Uključuje funkcionalnosti kao što su automatsko skaliranje, balansiranje opterećenja, sigurnosne karakteristike (WAF - Web Application Firewall) i SSL terminaciju. Omogućava optimizaciju performansi i zaštitu aplikacija.

2.2.3. Azure Storage

Implementacija Azure Storage bi bila korisna u daljim razvojima aplikacije u kojima će u porudžbinama biti prisutna slika.

Azure Storage je širok spektar usluga za skladištenje podataka koje nudi Microsoft Azure. Ova platforma omogućava korisnicima da čuvaju i upravljaju podacima u cloudu na siguran, skalabilan i visoko dostupni način. Azure Storage podržava različite tipove podataka, uključujući strukturisane, polustrukturirane i nestrukturirane podatke.

3. SPECIFIKACIJA

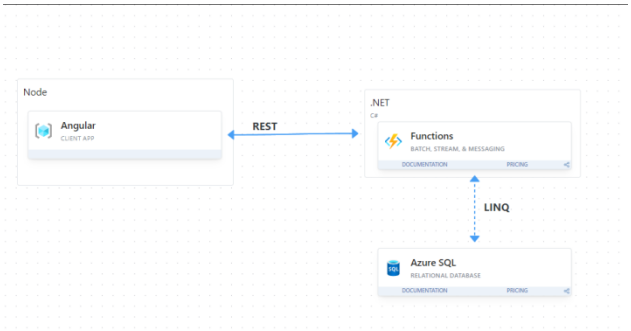
3.1. Nefunkcionalni zahtevi

U implementaciji fokus je stavljen na sledeće aspekte:

- Migracija na bezserversku arhitekturu putem *Azure* servisa, omogućavajući skalabilnost i minimiziranje infrastrukturnog održavanja
- *DevOps Pipeline* za *ASP.NET[2]* servis koji ostaje van bezserverskog okruženja, uz automatski *CI-CD* proces za izgradnju, testiranje i isporuku
- Monitoring i logovanje preko *Azure-a*, obezbeđujući uvid u performanse, stabilnost i sigurnost sistema

3.2. Arhitektura sistema

U ovom poglavlju detaljnije je opisana arhitektura sistema i tehnologije korišćene za razvoj istog. Softverski alat za podršku rada cvećare implementiran je kao veb-aplikacija koja se sastoji iz klijentskog i bezserverskog dela. Klijentski delovi navedene aplikacije implementirani su koristeći programski jezik TypeScript[3], a kao radni okvir odabran je Angular[4, 5], verzije 12. Centralni deo aplikacije implementiran je upotrebom bezserverskih tehnologija, funkcije su pisane u programskog jeziku C#[1] , korišćenjem radnog okvira .NET[2] verzije 6.0. Komunikacija u aplikaciji, vrši se preko HTTP protokola.



Node.js je neophodan za postavljanje razvojnog okruženja, upravljanje zavisnostima i olakšavanje procesa izgradnje i testiranja Angular[4] aplikacija. Iako se Angular[4] aplikacije izvršavaju na klijentskoj strani, Node.js pruža infrastrukturu i alate koji su ključni za njihov razvoj.

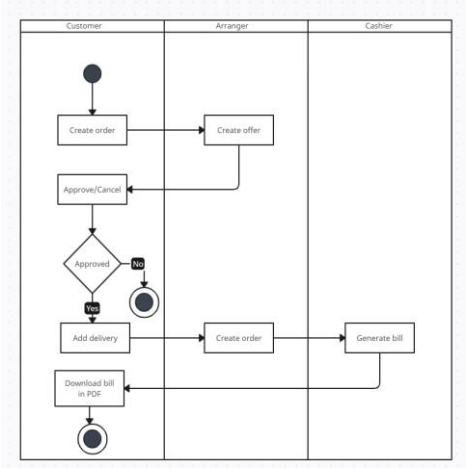
Angular[4] koristi HttpClient modul za komunikaciju sa backend-om putem RESTful API-ja. Zahtevi se šalju koristeći različite HTTP metode, a podaci se obično razmenjuju u JSON formatu. Korišćenje Observable-a omogućava asinkrono rukovanje podacima, dok RxJS operateri olakšavaju obradu odgovora i upravljanje greškama.

Azure Function App[8] omogućava brzo razvijanje i izvođenje funkcija koje se aktiviraju na osnovu različitih događaja. Kroz podršku za različite okidače, povezivanje sa drugim Azure servisima i mogućnost automatske skalabilnosti.

Azure SQL nudi fleksibilna i skalabilna rešenja za upravljanje relacionim bazama podataka u cloud-u. Sa visokom dostupnošću, sigurnosnim funkcijama, jednostavnom migracijom i snažnim alatima za monitoring.

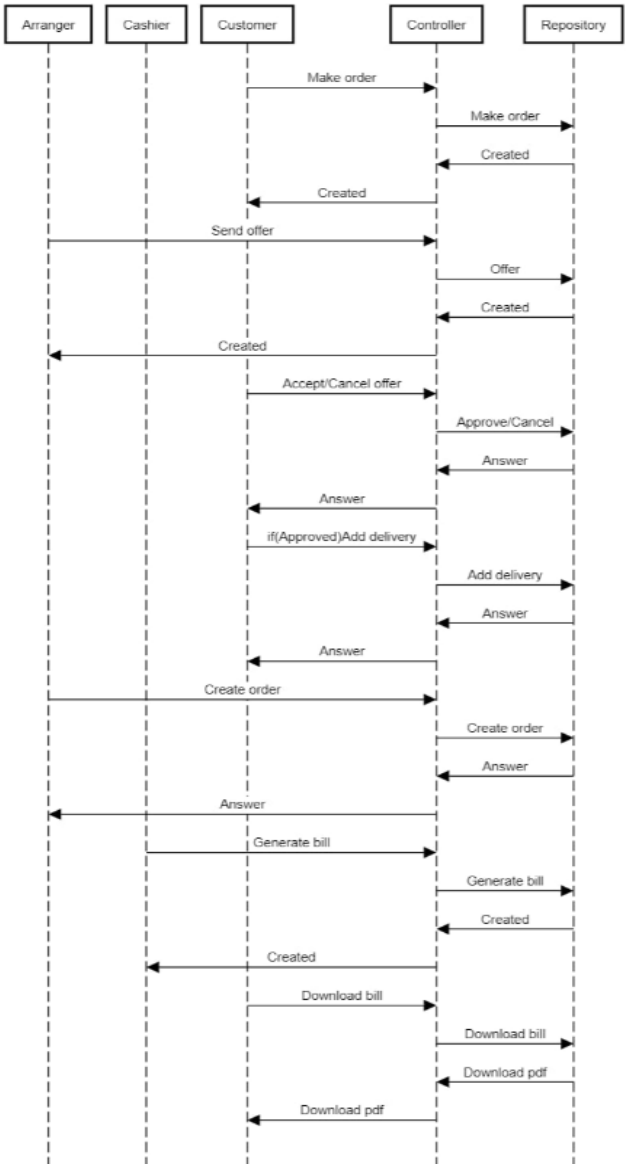
3.3. Activity diagram

Activity diagram je vrsta dijagrama u Unified Modeling Language (UML) koja se koristi za prikazivanje toka aktivnosti ili procesa unutar sistema. Ovi dijagrami su posebno korisni za modeliranje poslovnih procesa, radnih tokova i operacija u sistemu. Pomažu u razumevanju kako se aktivnosti međusobno povezuju i u kom redosledu se izvršavaju.



3.3. Sequence diagram

Sekvencni dijagram (sequence diagram) je jedan od najvažnijih dijagrama u Unified Modeling Language (UML) i koristi se za modeliranje interakcija između objekata ili komponenti u sistemu tokom vremena. Ovaj dijagram prikazuje kako objekti komuniciraju jedni s drugima putem slanja poruka, kao i redosled tih interakcija.



U datim dijagramima prikazana je komunikacija tri uloge unutar sistema (kupac, aranžer i kasir) kao i hronološko izvršavanje naredbi od strane svakog od njih.

Dijagrami su crtani u alatu Power Designer[6].

3.4. OPIS REALNOG SISTEMA

U ovom poglavlju opisano je pet klasa korisnika koje su od važnosti za ovaj softverski paket, kao i funkcionalni zahtevi koje ispunjava cvećara.

U realnom sistemu cvećare postoje sledeće klase korisnika koje su od interesa:

1. **Kupac** – poručuje aranžmane za koje može, a ne mora da zahteva dostavu.
2. **Aranžer** – pravi aranžmane kao i ponude na zahtevanu porudžbinu od strane kupca. Postavlja cene cveća i materijala.
3. **Šef** – njegova uloga sadrži mogućnost uvida u godišnju zaradu putem grafikona razdvojenog po mesecima i po godinama. Može dodeljivati povišice i smanjenja radnicima, naručiti robu od određenog dobavljača.
4. **Kasir** – ima mogućnost ispisivanja računa.
5. **Dobavljač** – izvršava dostave i ima uvid u sve dostave koje su poručene od strane poslovnica.

Karakteristike klasa korisnika informacionog sistema su:

1. **Neregistrovani korisnik i kupac** – poznavanje rada na računaru ovih uloga varira od lošeg do odličnog s obzirom da korisnici variraju od mlađih ka starijim, od edukovanih do needukovanih... Treba im omogućiti forme koje će posao raditi same, u smislu jednostavnih inputa ili dugmadi što je u našem sistemu i omogućeno.
2. **Aranžer** – očekuje se da osoba koja pripada ovoj grupi korisnika poseduje domensko znanje, kao i iskustvo aranžiranja u cvećari. Poznavanje rada na računaru spram informacionog sistema treba da bude poznato, s tim da se aranžer sa tim poslovima susretao i pre razvoja aplikacije. Ovoj grupi korisnika potrebno je što više olakšati posao jednostavnim formama za unos podataka u sistem, kako bi se više koncentrisali na razvoj proizvoda nego na ispravljanje bespotrebnih grešaka. Kako bi se sprečio nastanak grešaka pri unosu mnogobrojnih podataka u sistem, dosta pažnje je posvećeno validaciji podataka u toku unosa od strane prodavca.
3. **Šef** – pretpostavlja se da šef poseduje znanje rada na računaru, kao i domensko znanje, s obzirom da se bavi nabavkama i proračunima. Samim tim, nije nužno postavljanje jednostavnijeg korisničkog interfejsa, ali je poželjno da niko ne mora da se muči.
4. **Kasir i dobavljač** – za ove dve uloge se očekuje da imaju osrednje znanje rada na računaru, pogotovo kasir, kojoj je i posao računar (kasa). Omogućen im je jednostavan interfejs sa kojim ne bi trebali imati problema.

4. ZAKLJUČAK

U ovom radu analiziran je proces migracije postojećeg sistema sa monolitne arhitekture na bezserversku infrastrukturu, sa ciljem povećanja skalabilnosti, smanjenja troškova i optimizacije upravljanja resursima. Monolitna arhitektura, iako veoma jednostavna za razvoj, laka za testiranje I debagovanje sobzirom da su sve komponente unutar jednog paketa, omogućava direktnu komunikaciju, nailazi na probleme sa povećanjem broja korisnika. Neki od problema su održavanju infrastructure I upravljanju resursima. To je dovelo do prelaska na arhitekturu koja omogućava lakše prilagođavanje ovim problemima.

Migracija na Azure arhitekturu koja uključuje Azure funkcije kao i Azure SQL bazu podataka doprinela je poboljšavanju performansi, kao i donela mogućnost automatskog skaliranja aplikacije u situacijama kada to postaje potrebno.

Azure funkcije su funkcije s „hladnim pokretanjem“ (*cold start*) što takođe doprinosi upravljanju resursima, i kontroli troškova, jer se instance pokreću samo kada su potrebne. Ovo smanjuje troškove, jer korisnici plaćaju samo za vreme izvršenja. Prednosti im je takođe i fleksibilnost i skalabilnost jer omogućavaju lako skaliranje aplikacija prema potrebama korisnika bez potrebe za unapred definisanjem kapaciteta.

Korišćenje Azure arhitektura omogućava dalju intergraciju sa ostalim Azure servisima koji su navedeni iznad.

4. LITERATURA

[1] C# Language Documentation, https://learn.microsoft.com/en-us/dotnet/csharp/

[2] What is .NET, .NET Documentation, https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet

[3] TypeScript Documentation, https://www.typescriptlang.org/docs/

[4] Angular Documentation, https://v17.angular.io/docs

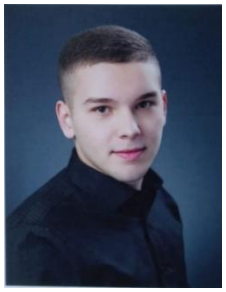
[5] Azure SQL Database, https://learn.microsoft.com/en-us/azure/azure-sql/database/sql-database-paas-overview?view=azuresql

[6] PowerDesigner, https://www.powerdesigner.biz/EN/powerdesigner/lp-data-modeling-powerdesigner-trial-source_adwdatamodel.html?gad_source=1&gbraid=0AAAAAocL3fkVU5cOzdN78FR0PPREp7_vG&gclid=Cj0KCQjw05i4BhDiARIsAB_2wfBNjhs4Khim-AHrd_nzc2--T2zSi4Jo1p9iqSTtUT5qP5WWM7ocd9kaAs8SEALw_wcB

[7] Azure Documentation, https://learn.microsoft.com/en-us/azure/?view=azuresql&product=popular

[8] Azure Functions Overview, https://learn.microsoft.com/en-us/azure/azure-functions/functions-overview?pivots=programming-language-csharp

Kratka biografija:



Mihajlo Savić rođen je 05.08.2000. godine u Novom Sadu. Godine 2019. upisao je Fakultet Tehničkih Nauka u Novom sadu, odsek Računarstvo i automatika. Osnovne studije završio je u septembru 2023. Od Oktobra 2023. upisuje Master akademske studije.
kontakt: justdoit0508 @gmail.com