Opravdanost implementiranog rešenja

PSP treba da bude loosely-coupled sa veb-prodavnicom

PSP treba da ima API koji je prilagođen radu sa raznim sistemima i prodavcima, od sistema koji imaju jednog prodavca (npr. klasična veb-prodavnica) do sistema koji uslužuje više prodavaca. Prvi najbitniji zadatak ove tačke jeste pametno dizajniranje ove interakcije.

Implementacija

- PSP prodavcima nudi mogućnost integrisanja njihove prodavnice sa uslugama koje PSP nudi
  - Neophodno je da se prodavac registruje na PSP-u tako što pruža osnovne informacije o svojoj prodavnici
  - Prilikom registracije, prodavac dobija token jedinstven za njegovu prodavnicu, koji treba da upotrebi kada šalje zahteve PSP-u iz svoje veb prodavnice
  - Za prodavca se vezuju različiti načini plaćanja koje je prodavac odabrao na svom nalogu koji je kreirao na PSP-u i njih je moguće utvrditi preko tokena kada PSP-u stigne zahtev za plaćanje određene robe
- Veb prodavnica treba da poseduje mehanizam za uspostavljanje komunikacije sa PSP-om
  - Prodavcu se nudi mogućnost saradnje sa PSP-om tako što može uneti token koji je dobio pri registraciji na PSP-u
  - Kada kupac želi da plati odabrane proizvode, PSP-u se šalje zahtev zajedno sa tokenom koji je prodavac obezbedio, tako da je moguće utvrditi na čije naloge se novčana sredstva trebaju prebaciti i koje metode plaćanja ponuditi kupcu

Opravdanost odabranog rešenja

- PSP ne pravi nikakve pretpostavke o vrsti proizvoda koju određeni prodavac nudi
- PSP ne pravi nikakve pretpostavke da li klijent koji šalje zahtev predstavlja jednu veb prodavnicu ili sistem koji podržava veliki broj različitih prodavnica
- PSP samo propisuje pravila neophodna za uspešno uspostavljanje komunikacije (prodavnica mora biti autentifikovana preko svog tokena kako bi uspešno koristila usluge)
- Kao klijent usluga PSP-a implementirana je e-commerce platforma koja podržava različite prodavnice i prodavcima nudi mogućnost integracije sa PSP-om

PSP treba da bude plagabilan

PSP treba da bude plagabilan, gde svaki vid plaćanja predstavlja jedan plug-in. PSP treba projektovati tako da se što jednostavnije mogu podržati novi servisi za plaćanje (Payoneer, druge kriptovalute itd).

Implementacija

- Svaki način plaćanja predstavlja odvojen servis koji autonomno vrši pretplatu korisnika na taj način plaćanja, njegovo odjavljivanje i nudi klijentima mogućnost plaćanja tom metodom plaćanja
- Kada se kreira novi servis za plaćanje, neophodno je proširiti API Gateway novim pravilima za rutiranje, kao I frontend aplikaciju koja će krajnjem korisniku nuditi mogućnost interakcije sa tim načinom plaćanja (prijavljivanje, unos neophodnih podataka ..)

Opravdanost odabranog rešenja

- Način plaćanja predstavlja plug-in time što je odvojen mikroservis u aplikaciji
- Novokreirani servis za plaćanje ne utiče na rad postojećih servisa, tačnije, uvođenje novog načina plaćanja ne zahteva nikakvu intervenciju nad implementacijom postojećih servisa za plaćanje niti ometa njihov dotadašnji rad
- Postojeće rešenje se proširuje u samo dve tačke, API gateway (kako bi zahtevi klijenata bili ispravno prosleđeni novom servisu) i frontend aplikacija (kako bi klijent mogao interagovati sa novim načinom plaćanja)

- ***PSP treba da ima arhitekturu koja podržava visoku dostupnost***

High-availability arhitektura podržava jednostavno skaliranje sistema. Integracija sa novim prodavcem (tipa veb-prodavnica) ili novim načinom plaćanja treba da se omogući bez gašenja PSP-a.

Implementacija

- Aplikacija je implementirana prema pravilima mikroservisne arhitekture
- Moguće je podizanje više instanci servisa koji je pod opterećenjem
- Podignut je registar servisa kom se svaki servis javlja pri svom pokretanju
- API Gateway i servisi koji žele da pošalju zahtev nekom servisu obraćaju se registru kako bi utvrdili na koju adresu treba poslati zahtev
- API Gateway igra i ulogu load balancer-a kada preusmerava pristigle zahteve klijenata

Opravdanost odabranog rešenja

- Dostupnost sistema povećava se podizanjem više instanci servisa jer je moguće obraditi veći broj pristiglih zahteva od strane klijenta
- Problem adresiranja servisa koji se pokreću i gase po potrebi rešen je uvođenjem servis registra
- Usvajanjem mikroservisne arhitekture, omogućena je fleksibilnost u skaliranju, tako da je moguće pokrenuti veći broj instanci samo u tom trenutku opterećenog servisa, ne i svih ostalih delova sistema
- Load balancing koji obavlja API gateway osigurava da instance istog servisa budu što ravnomernije opterećene, kako bi skaliranje instanci dovelo do željenog efekta
- Integracija PSP-a sa novom veb prodavnicom ne dovodi do potrebe za gašenjem PSP-a jer se bazira na registraciji novog korisnika na sistem koji je dužan da se predstavi PSP-u u zahtevima za plaćanje koje će mu upućivati (prodavnica se

povezuje upotrebom postojećih usluga koje PSP pruža, nezavisno od toga da li neki određeni prodavac želi da se integriše ili ne)

- Dodavanje novog načina plaćanja uzrokuje potrebu za novom verzijom API gateway-a I frontend aplikacije, a neprekinuta dostupnost klijentima može se postići nekom od tehnika zero downtime deployment-a