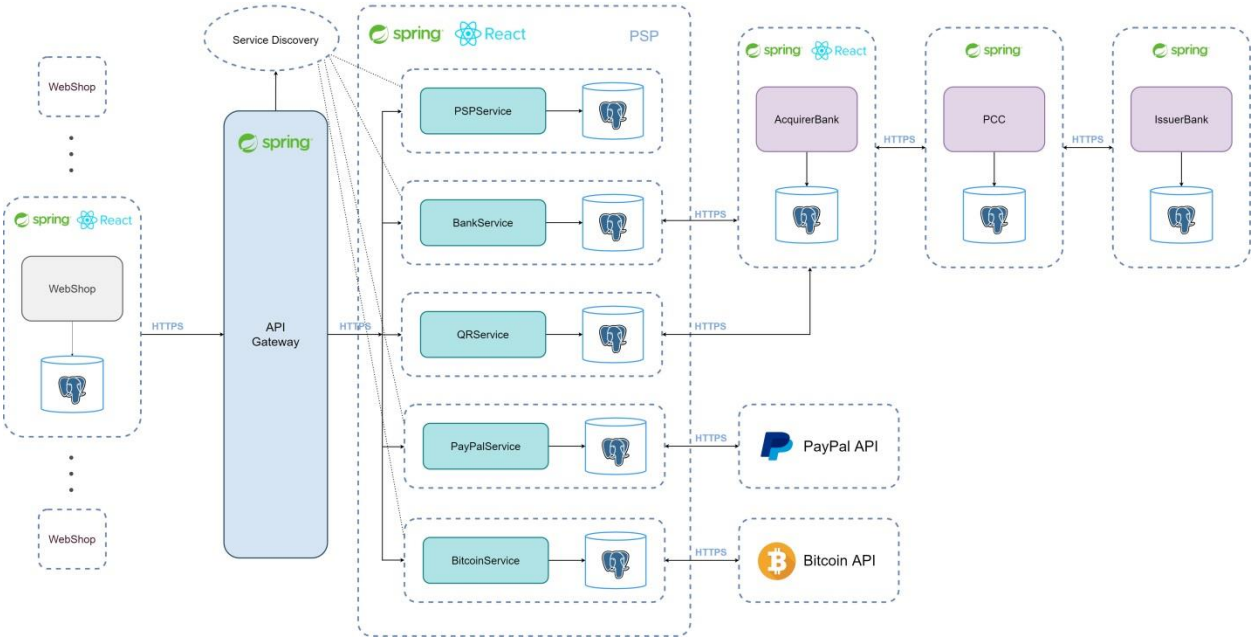
Opis osobina PSP-a

Arhitektura sistema

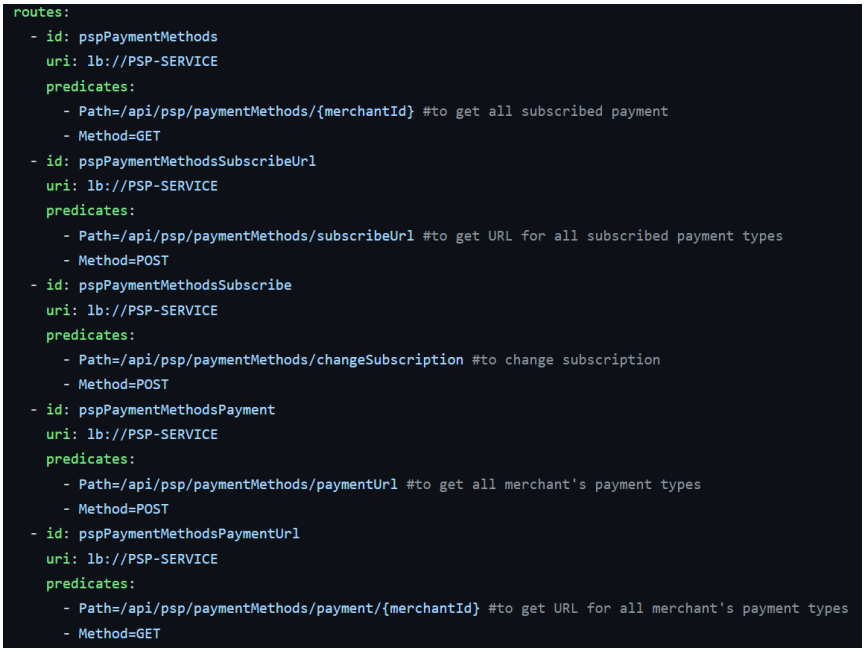


Slika 1. Arhitektura sistema

Osobine PSP-a

PSP treba da bude loosely-coupled sa veb-prodavnicom

PSP poseduje *API gateway* koji nudi *REST end-point*-e preko kojih je moguće pristupiti internim mikroservisima, a najčešće *PSPService*-u preko kog ide najviše zahteva. *API gateway* i svi servisi se registruju kod *service discovery*-a kao klijenti pomoću *@EnableEurekaClient* anotacije u *main* klasi. Moguće je proslediti zehtev i pristupiti mikroservisima na osnovu imena i portova registrovanih klijenata u *servis discovery*-u. U *API gateway*-u nije potrebno eksplicitno navoditi port mikroservisa već ime pod kojim je registovan. (slika 2.)



Slika 2. *API Gateway*

PSP treba da bude plagabilan

PSP poseduje mikroservisnu arhitekturu gde svaki servis predstavlja zaseban način plaćanja. Osim ovih servisa, dodatno sadrži i *PSPService* preko kojeg se odvijaju gotove sve funkcionalnosti vezane za samo plaćanje. Ovaj servis svestan je svih mogućih načina plaćanja koje poseduje PSP i poseduje mogućnost preusmeravanja ka njima. Ovo je realizovano na taj način što se u bazi podataka *PSPService*-a, čuvaju svi mogući načini plaćanja odnosno mikroservisi sa svojim nazivima prethodno definisanim u *spring.application.name*-u. Primeri takvih preusmeravanja bi bili pri odabiru načina plaćanja od strane klijenta i promeni pretplate na određeno plaćanje od strane administratora.

Primer jednog toka plaćanja preko banke bi bio sledeći:

Klijent klikne da želi da obavi kupovinu, preko *API gateway*-a se preusmeri zahtev na *PSPService*, koji potom otvori klijentu svoj *front* sa podržanim načinima plaćanja. Klijent odabere plaćanje preko banke i *PSPService* na osnovu tog odabira, preusmeri zahtev ka *BankService*-u, a koji potom ,,pogodi'' određeni *end-point acquirer* banke.

Prilikom dodavanja novog servisa za plaćanje, bilo bi potrebno uneti u bazu podataka njegov jedinstveni identifikator, naziv, opis, kao i jedinstveni naziv samog servisa (definisanog pod *spring.application.name* u *application.properties* fajlu). Kako bismo izbegli ovo ručno dodavanje, mogli bismo da generišimo formu koju bi administrator PSP-a popunio sa ovim podacima, a jedinistveni identifikator bi bio automatski generisan (*UUID*). Još jedna stavka koju bi novi servis za plaćanje morao da zadovolji je da obezbedi, u ovom slučaju, *POST* putanje */createTransaction* i */checkIfMerchantExists* kako bi PSP mogao automatski da prosleđuje kreiranje transakcija i promenu pretplate ka novom servisu. Takođe, morao bi da se preko *@EnableEurekaClient* anotacije u *main* klasi i dodatnih podešavanja u *application.properties* fajlu registruje kod *Eureka Service Discovery*-a, kako bi bio vidljiv ostalim mikroservisima.

Još jedna stavka koja bi mogla da olakša plagabilnost PSP-a je da generišemo prethodno opisan način dodavanja servisa u *.dll* biblioteku, čijim bi korišćenjem automatski dobili potrebne, gore navedene metode.

```
log.info("Trying to redirect to " + paymentMethodType.getServiceName());
httpResponse = restTemplate.postForObject("https://" + paymentMethodType.getServiceName() + "/createTransaction", createTransactionDTO, TransactionResponseDTO.class);
log.info("Successfully redirected to " + paymentMethodType.getServiceName());
```

Slika 3. Preusmeravanje zahteva ka odobaranom načinu plaćanja

PSP treba da ima arhitekturu koja podržava visoku dostupnost

Novi prodavac koji želi da koristi usluge PSP-a, trebalo bi da se registruje putem dodeljene forme, kako bismo imali njegove podatke prilikom plaćanja ili pretplate. Pomoguću *end-point*-a koje pruža *API Gateway*, moći će da koristi sve usluge PSP-a.

Upotreba novog načina plaćanja se realizuje putem dodavanja novog mikroservisa u PSP, a čija upotreba zahteva određene stvari opisane u prethodnom potpoglavlju.

Ni u jednom ni u drugom slučaju nije potrebno gašenje već postojećih mikroservisa koji pripadaju PSP-u, kao ni gašenje *API gateway*-a ili *Eureka Service Discovery*-a.