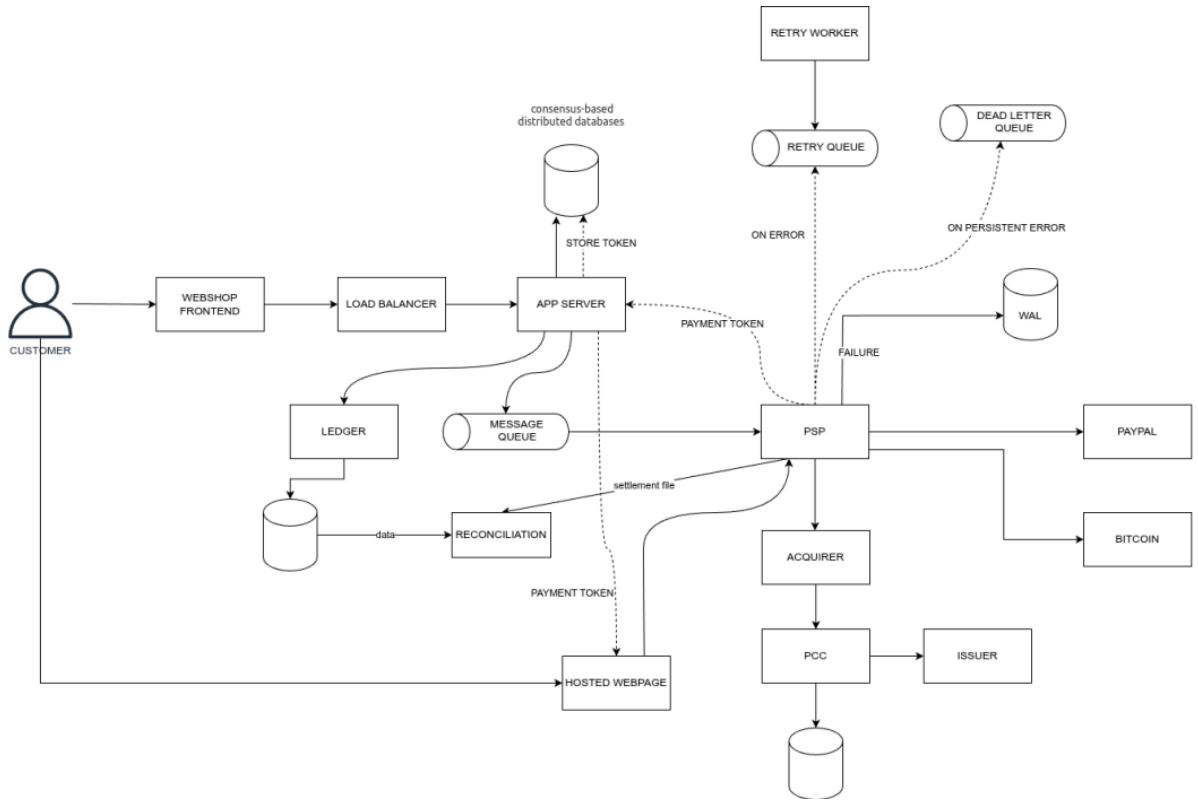
Dizajn sistema

R2 7/2024 Nikola Savić • R2 8/2024 Jovan Najdovski

Slika sistema



Tehnologije

Servisi

- **Go (Golang)** je odabran zbog svoje efikasnosti u rukovanju paralelnim zahtevima, što je ključno za sistem plaćanja sa visokim opterećenjem. Dobijamo smanjivanje latencije, dok ugrađene biblioteke olakšavaju izgradnju stabilnih i skalabilnih mikroservisa. Pored toga, odlična podrška za REST API-je omogućava jednostavnu integraciju sa PSP-om i ostalim komponentama sistema. Koristiće se Gin framework.

Load Balancer

- **NGINX** je odabran zbog svoje sposobnosti da efikasno raspoređuje saobraćaj između mikroservisa. U ovom sistemu, koji zahteva nisku latenciju i visoku dostupnost, NGINX pruža potrebne performanse i pouzdanost. Njegova podrška za SSL/TLS omogućava sigurnost u razmeni podataka između korisnika, agencije i PSP-a. Pored toga, omogućeno je i dodatno keširanje.

Service Discovery

- **Consul** je izabran zbog svoje mogućnosti automatskog otkrivanja servisa i integrisanih health-check funkcionalnosti. Ovo je ključno u sistemu plaćanja sa velikim brojem mikroservisa, jer omogućava jednostavno povezivanje između servisa i garantuje kontinuitet rada čak i u slučaju grešaka.

Baza podataka

- **PostgreSQL** je odabrana kao standard sql baza podataka za sve mikroservise sistema.

- **(Opciono kao dodatak - Distribuirana baza) CockroachDB** je odabran zbog svoje sposobnosti automatske replikacije podataka i horizontalnog skaliranja. U sistemu za plaćanje, gde je dostupnost i konzistentnost ključna, CockroachDB osigurava otpornost na greške i kontinuitet rada.

Skaliranje i kontejnerizacija

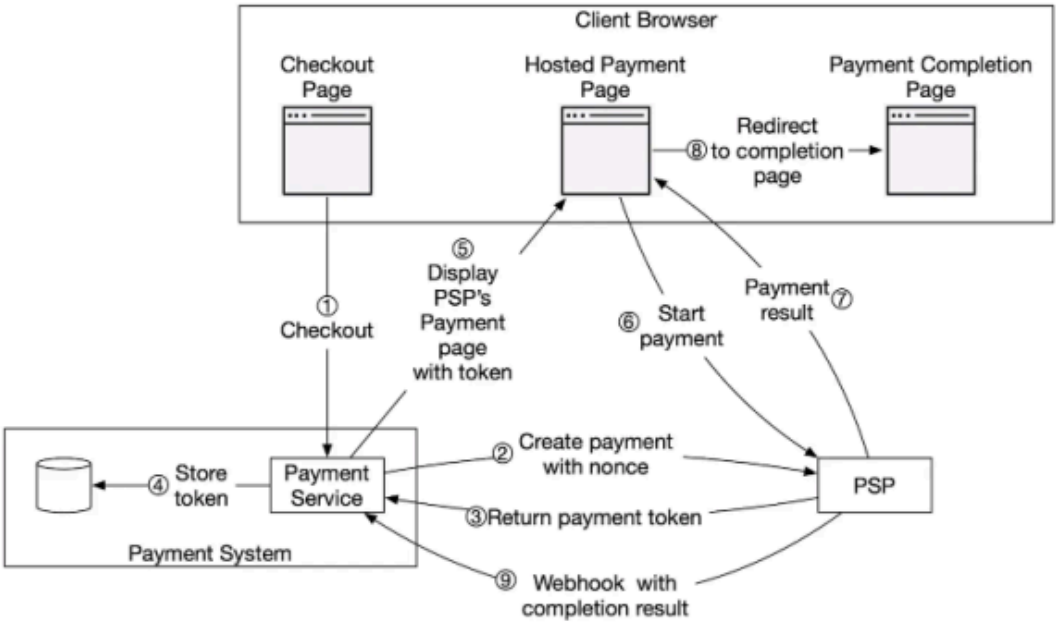
- **Docker -** Docker je izabran jer omogućava jednostavnu kontejnerizaciju servisa i njihovo skaliranje prema potrebama sistema. U slučaju povećanog saobraćaja, Docker omogućava brzo pokretanje novih instanci, čime se osigurava visoka dostupnost i optimizacija resursa.

Tok podataka

Tok podataka između agencije i Payment Service Provider-a (PSP) se obavlja kroz sledeće korake:

1. Kada korisnik završi sa odabirom vozila i klikne na dugme za potvrdu iznajmljivanja u mobilnoj ili web aplikaciji, pokreće se proces obrade plaćanja. Aplikacija šalje potrebne informacije backend servisu agencije za iznajmljivanje vozila, uključujući podatke o izboru korisnika (iznos, valuta, odabrano vozilo…) i podatke o samom korisniku (ime, email adresa, telefonski broj…).
2. Nakon što backend servis agencije primi podatke o plaćanju od web ili mobilne aplikacije, šalje zahtev za registraciju plaćanja PSP-u. Ovaj zahtev sadrži podatke kao što su ID prodavca, Password prodavca, iznos, valuta, vreme transakcije, vreme isteka (kada zahtev za plaćanje ističe) i jedinstveni identifikator iznajmljivanja, koji se koristi za praćenje tog iznajmljivanja i osigurava da se registracija plaćanja obavi samo jednom.
3. PSP generiše i vraća jedinstveni token, koji predstavlja identifikator registracije plaćanja sa strane PSP-a. Ovaj token omogućava kasniju proveru statusa transakcije i praćenje njenog toka kroz celi proces.
4. Kada primi token, backend servis prodavnce ga čuva u svojoj bazi podataka kako bi mogao kasnije da prati status plaćanja.
5. Korisnik se preusmerava na PSP-ovu hostovanu stranicu za plaćanje, koja zahteva ključne podatke. Jedan od podataka je token (dobijen u koraku 3), koji omogućava stranici da pristupi važnim podacima o transakciji, kao što je iznos za uplatu.

6. Korisnik na PSP-ovoj hostovanoj stranici unosi podatke o plaćanju (broj kartice, datum isteka, CVV, itd.). Nakon što klikne dugme za plaćanje, ovi podaci se šalju PSP backend servisu, koji zatim pokreće proces obrade transakcije.
7. Nakon završetka transakcije, PSP vraća status transakcije (uspešna, neuspešna, na čekanju).
8. Nakon što se status plaćanja dobije, korisnik je preusmeren na URL za preusmeravanje. Ovo je stranica na kojoj korisnik može videti rezultat plaćanja (uspešno ili neuspešno).
9. PSP šalje obaveštenje putem webhook-a na URL registrovan tokom inicijalne konfiguracije sa backend servisom agencije. Ovaj webhook sadrži informacije o statusu plaćanja (uspešno, neuspješno ili na čekanju), kao i dodatne detalje o transakciji. Ovaj korak se obavlja asinhrono.



Podaci koji se razmenjuju tokom procesa plaćanja su sledeći:

1. PSP –> Acquirer (bank)
    - Merchant ID
    - Merchant password
    - Amount
    - Merchant order ID
    - Merchant timestamp

2. Acquirer (bank) –> PSP
    - Payment URL
    - Payment ID

3. Acquirer (bank) –> PCC

- Acquirer order ID
- Acquirer timestamp
- PAN
- Security code
- Card holder name
- Amount

4. PCC –> Issuer (bank)
    - Acquirer order ID
    - Acquirer timestamp
    - PAN
    - Security code
    - Card holder name
    - Amount

5. Issuer (bank) –> PCC
    - Status
    - Acquirer order ID
    - Acquirer timestamp
    - Issuer order ID
    - Issuer timestamp

6. PCC –> Acquirer (bank)
    - Status
    - Acquirer order ID
    - Acquirer timestamp
    - Issuer order ID
    - Issuer timestamp

7. Acquirer (bank) –> PSP
    - Status
    - Acquirer order ID
    - Acquirer timestamp
    - Payment ID
    - Merchant order ID

Skladištenje podataka

Servis agencije

Tabela narudžbi

- UUID
- Amount
- Currency
- User ID
- Vehicles Details

Tabela tokena

- UUID
- PSP Token
- Expires at

PSP

Tabela transakcija

- Transaction ID
- Merchant ID
- Merchant Order ID
- Status (in progress, successful, failed)
- Timestamp
- Merchant_timestamp
- Amount
- Currency

Tabela merchant-a

- Merchant ID
- Hashed merchant password
- Salt
- SuccessURL
- FailedURL
- ErrorURL

Tabela subscription-a

- Merchant ID
- Subscription ID
- Subscription method

Bank

Tabela o korisnicima

- User ID
- Name
- Surname
- Birthday
- Email
- Phone number

Tabela o računima

- Account ID
- User ID
- Balance
- Currency
- Date created (opened)
- Status (active, closed, blocked)

Tabela o karticama

- Card ID

- User ID
- Card number
- Expiry date
- Card type (debit, credit, prepaid)
- Is tokenized

Tabela o transakcijama
- Acquirer order ID
- Payment ID
- Merchant ID
- Amount
- Currency
- Timestamp
- Status
- Card number – poslednje 4 cifre

Tabela merchant-a
- Merchant ID
- Bank account ID

Bank Gateway

Tabela transakcija
- Transaction ID
- Merchant ID
- Merchant Order ID
- Routed Bank ID
- Status (successful, failed)
- Timestamp
- Amount
- Currency

Tabela banaka
- Merchant ID
- Bank ID

PCC

Tabela transakcija
- Transaction ID
- Status (successful, failed)
- Timestamp
- Acquirer ID
- Issuer ID

API

Login:

POST http://localhost:8080/login

Body (json)

```
{
  "username": "username",
  "password": "password"
}
```

Register:

POST http://localhost:8080/register

Body (json)

```
{
  "name": "name",
  "email": "john.doe@mail.com",
  "username": "username",
  "password": "password"
}
```

Get vehicles:

GET http://localhost:8080/vehicles

Request Headers: Authorization

Get vehicle by id:

GET http://localhost:8080/vehicle/:id

Request Headers: Authorization
Path Variables: id

Create vehicle:

POST http://localhost:8080/vehicles

Request Headers: Authorization
Body (json)

```
{
  "id": 5,
```

```
  "category": "SUV",
  "name": "Reno Capture",
 "description": "Designed for students, this affordable car includes
   reliable motor and good services to support different
   activities.",
  "price": 3200
}
```

Purchase

POST http://localhost:8080/vehicle/purchase

Request Headers: Authorization
Body (json)

```
{
  "vehicleId": 1,
  "days": 1
}
```

Get user active payments

GET http://localhost:8080/user/info

Request Headers: Authorization