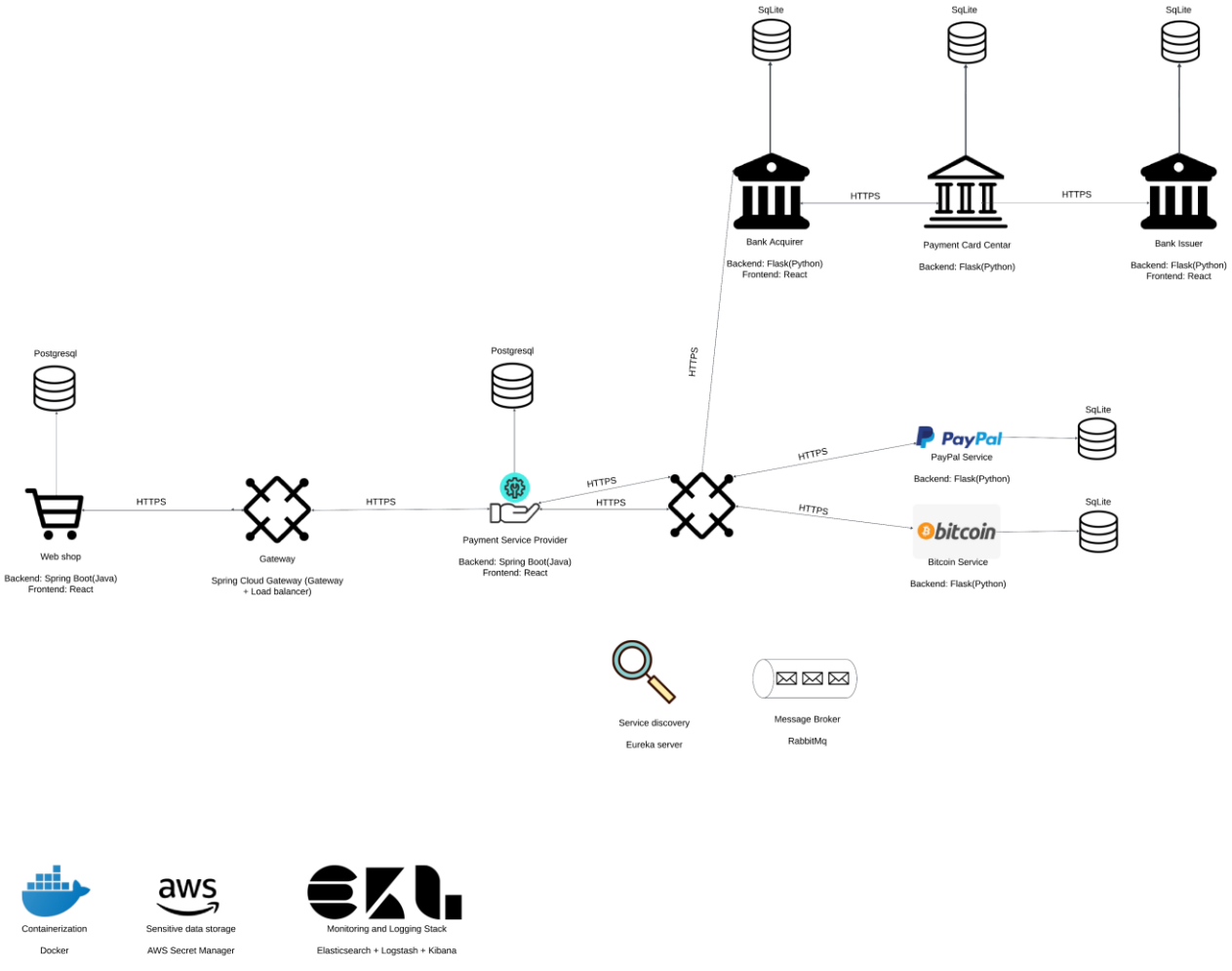
DIZAJN SISTEMA



Slika 1 – dizajn sistema

Na slici 1 je prikazan dizajn sistema sa svim korišćenim tehnologijama.

1. Komponente sistema:

1. **Web shop:**
   Web aplikacija koja omogućava korisnicima da kupuju proizvode i pretplaćuju se na različite usluge.

2. **Payment Service Provider (PSP):**
   Web aplikacija koja omogućava korisnicima da biraju načine plaćanja koje će njihova kompanija da koristi (pretplati) npr. PayPal, Bitcoin ili Card...

3. **Gateway (web shop <-> PSP):**
   API Gateway koji se koristi za rutiranje zahteva između web shopa i PSP-a. Povezuje se sa Eureka serverom kako bi dobio dostupne instance PSP-a i

implementira Round Robin load balancing kako bi ravnomjerno raspoređivao saobraćaj među njima. Takođe, vrši Rate limiting kako bi kontrolisao broj zahteva, čime se poboljšava performansa i zaštita sistema. Ovaj pristup omogućava da se oba servisa razvijaju i skaliraju nezavisno, ostvarujući loosely coupling.

4. **Gateway (PSP <-> PMS):**
   API Gateway koji se koristi za rutiranje zahteva između PSP-a i drugih servisa za plaćanje. Povezuje se sa Eureka serverom kako bi dobio dostupne instance servisa i implementira Round Robin load balancing kako bi ravnomjerno raspoređivao saobraćaj među njima. Takođe, vrši Rate limiting kako bi kontrolisao broj zahteva, čime se poboljšava performansa i zaštita sistema.

5. **Service discovery:**
   Omogućava load balancing unutar PSP servisa tako što otkriva dostupne instance i omogućava efikasno usmeravanje saobraćaja.

6. **Message Broker (RabbitMQ):**
   PSP je pretplaćen na RabbitMQ kako bi bio obavešten o postojanju novih načina plaćanja. Novi servisi za plaćanje, nakon što se podignu, obaveštavaju sistem o svom postojanju slanjem poruka. Super admin ima mogućnost da odobri nove načine plaćanja. Na ovaj način se omogućava plagabilnost, odnosno dodavanje novih načina za plaćanje unutar PSP bez gašenja.

7. **Servisi za plaćanje:**
   Mikroservisi koji omogućavaju različite načine plaćanja kao što su PayPal, Bitcoin, i Banka (QR kod i kartica).

8. **Banka:**
   Banka prodavca i Banka kupca su dve instance iste aplikacije koje simuliraju plaćanje u bankama. Payment Card Center (PCC) je posrednik između banke kupca i banke prodavca, omogućavajući sigurnu i efikasnu obradu transakcija.

2. Nefunkcionalni zahtevi:

1. **Loosely Coupling:**
   Objašnjeno u poglavlju 1.3 i 1.4

2. **Plagabilnost:**
   Objašnjeno u poglavlju 1.6

3. **Visoka dostupnost sistema:**
   Korišćenje Docker kontejnera omogućava praćenje trenutne opterećenosti servisa, a na osnovu toga moguće je podizanje dodatnih instanci servisa kako bi

se osigurala visoka dostupnost i otpornost sistema. Takođe u tačkama 1.3 i 1.4 je objašnjeno kako se ostvaruje load balancing kada je podignuto više instanci nekog servisa.

4. **Zaštita podataka:**
   Za zaštitu podataka koristićete TLS za enkripciju prenosa između servisa, AES-256 za enkripciju osetljivih podataka u bazi, i BCrypt za sigurno hashovanje lozinki.

5. **Mere kontrole pristupa:**
   Za autentifikaciju i autorizaciju korisnika, koristićete OAuth2 protokol. Implementiran je RBAC kako bismo dodelili pristup na osnovu uloga (npr. administratori, korisnici, super admini) i omogućili detaljnu kontrolu prava pristupa.

6. **Monitoring:**
   ELK Stack (Elasticsearch, Logstash, Kibana): Koristiće se ELK Stack za prikupljanje, čuvanje i analizu logova svih pristupa mrežnim resursima i podacima o karticama. Logovi će biti analizirani pomoću Kibane za vizualizaciju i praćenje događaja.

3. Dodatno:

Bilo bi dobro koristiti AWS Secrets Manager za čuvanje API ključeva, Merchant ID-a, Marchant Passworda, lozinki i konekcija za bazu podataka, jer pruža sigurno čuvanje osetljivih podataka. Međutim, to se naplaćuje i nije izvodljivo u trenutnim uslovima, pa se te informacije trenutno čuvaju u application properties ili environment varijablama.