CASE TÉCNICO

TESTE PARA POSIÇÃO DE ARQUITETO

acesso bankly

FASE 1

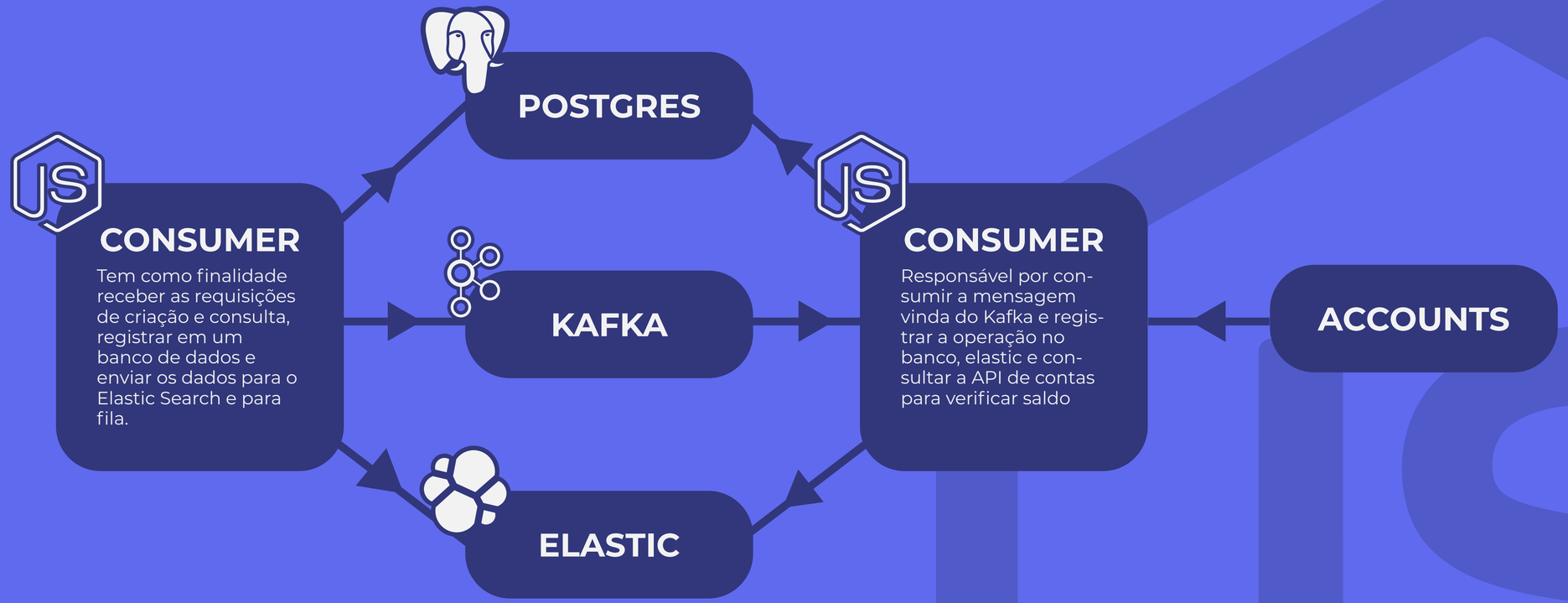
Cenário do case:
Criação de uma API para realização de transferência de valores entre duas contas, essa transferência **não precisará ser online.**
Para consultar contas, fazer movimentação financeira e consulta de saldo deve-se usar a API de conta que está documentada abaixo.
Deve haver um endpoint para consulta de status da transferência, que podem ser:

- In Queue
- Processing
- Confirmed
- Error

Nestes casos é necessário retornar o motivo do erro!
Temos que ter logs de todas as operações efetuadas pela API, sejam consultas ou transferências.
O tempo de resposta da API tem que ser baixo e temos que ter a menor quantidade de erros possível.

acesso bankly

POC DESENVOLVIDA



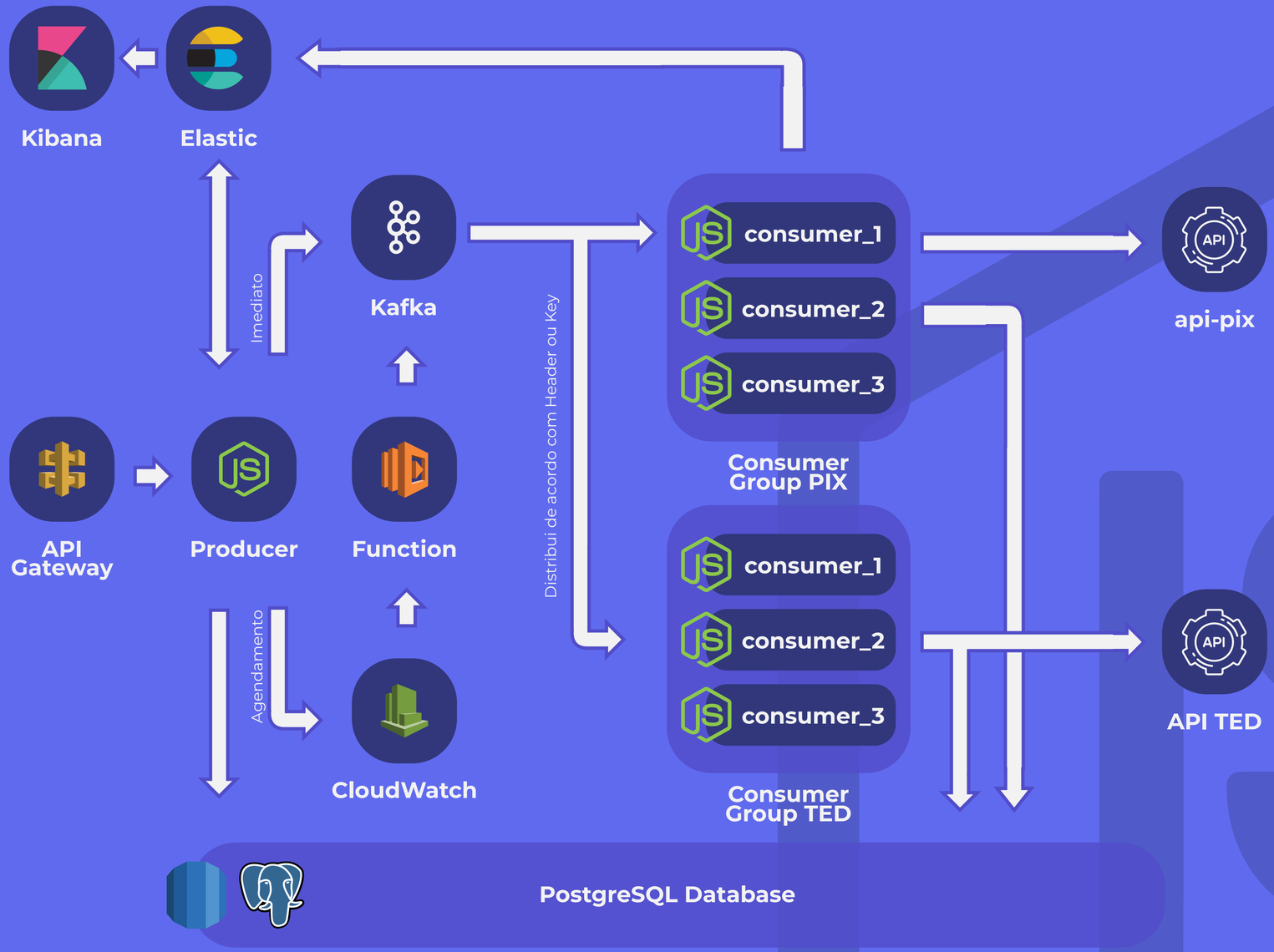
acesso bankly

FASE 2

Cenário do case:
Somos um Banking as a Service, e um dos produtos mais utilizados por nossos parceiros é são as APIs de PIX e TED. Recentemente o Banco Central determinou que o PIX deverá suportar agendamento de pagamentos, e queremos aproveitar que essa oportunidade para incluir também o agendamento de TED que é uma funcionalidade bastante requirida por nossos parceiros.
Ambas as formas de pagamento tem características distintas, por exemplo. O SLA de um PIX é de 40 segundos, enquanto o SLA de um TED é 2 horas. Tanto PIX, quanto TED podem ter agendamentos recorrente, exemplo todo dia 5, por até 12 meses.
Temos um prazo apertado para entregar o agendamento PIX, devido o calendário do BACEN, já o TED, por ser uma funcionalidade adicional, mas o time to market é importantíssimo, pois estamos perdendo oportunidades de novos negócio.
1. Qual a arquitetura de solução você criaria para suportar esta nova funcionalidade?
2. Quais tecnologias seriam utilizadas?
3. Como a solução é escalável?
4. As operações serão síncronas ou assíncronas?
5. Este for utilizar algum banco de dados, qual será utilizado?
6. Qual é a sua proposta de entrega para cumprir os prazos?

acesso bankly

ARQUITETURA PROPOSTA



acesso bankly

ARQUITETURA PROPOSTA

1. Qual a arquitetura de solução você criaria para suportar esta nova funcionalidade?
Vide esquema no slide anterior.

2. Quais tecnologias seriam utilizadas?
Optaria por utilizar uma arquitetura Cloud-Based com recursos oferecidos pela AWS pelo grande leque de ferramentas oferecida pela empresa. Serviços desenvolvidos em Node e Lambda com o auxílio de um Kafka gerenciado pela AWS para centralizar mensageria.

3. Como a solução é escalável?
Para que a solução se torne plenamente escalável, é necessário configurar as instâncias de serviços para que rodem em container com o auxílio de um EKS, tornando assim a aplicação passível de auto-scaling de acordo com a demanda. O uso de Kafka para mensageria também é crucial, porque possibilita o uso dos grupos de recursos, onde, dependendo da criticidade, é possível distribuir a mensagem em diversos consumers, aumentando a garantia de que a mensagem foi processada.

4. As operações serão síncronas ou assíncronas?
Assíncronas, garantindo uma resposta para o usuário dizendo que recebemos a solicitação dele e garantindo com base na arquitetura do sistema que a mensagem será processada como deve ser.

5. Este for utilizar algum banco de dados, qual será utilizado?
Optei por um PostgreSQL instanciado no RDS, para resguardar a disponibilidade do banco em um serviço utilizado mundialmente, e também porque apesar de um banco relacional oferecer uma indexição inferior a um banco não-relacional, no longo prazo, bancos não-relacionais tendem a oferecer um grande impacto de performance, principalmente quando se trabalha com transacional, como é o caso de transferências bancárias. De quebra, haverá um elastic search que servirá para dar um score aos registros “mais quentes” e proporcionará uma experiência melhor quando o usuário realizar uma consulta das transações.

6. Qual é a sua proposta de entrega para cumprir os prazos?
A proposta de entrega de MVP seria entregar sem a consulta de baixa latência, ou seja, sem a parte de Elastic, para garantir velocidade no desenvolvimento. Uma vez que as demais partes do software tratam de desenvolvimento de baixa complexidade, onde os consumers do grupo de TED e PIX devem ser muito parecidos (ao menos conceitualmente), e o producer é o core da aplicação, onde serão centralizadas as requisições, logo, é inviável abrir mão dele. Quantos aos serviços auto-gerenciaveis, como Kafka por meio do MSK e PostgreSQL por meio do RDS, não devem impactar a entrega, uma vez que tratam-se de instâncias de baixa complexidade para implementação, feitas quase que exclusivamente por meio de console da AWS.

acesso bankly