AS-IS (Situação Atual Avaliada pelo MCTI)

III. Integração IdWall para KYC da plataforma

Os desenvolvimentos tecnológicos experimentais foram orientados à **construção de novos mecanismos** para os processos de análise e validação cadastral de usuários, objetivando a **ampliação da capacidade sistêmica de captação, processamento e disponibilização de dados** dos usuários na plataforma **Superbid Webservice**, visando minimizar a incidência de possíveis ações fraudulentas.

Antes do projeto:

- O mecanismo de segurança existente para a plataforma envolvia apenas uma integração com **bureau externo**, que **dispunha parcialmente de dados cadastrais**.
- Havia **limitação na quantidade e tipo de informações disponíveis**, o que resultava em **baixa efetividade na identificação dos usuários**.
- O armazenamento dos dados era **on-premisse**, gerando **altos custos de manutenção**, limitação de buffer e restrições de capacidade.

Riscos Tecnológicos e Incertezas Superadas

Devido à **natureza experimental do projeto**, havia **riscos de insucesso** na integração com a plataforma externa IdWall, incluindo:

- **Incompatibilidade tecnológica** entre os sistemas da Superbid e a IdWall, exigindo soluções de adaptação.
- **Intermitência e instabilidade** na comunicação entre os ambientes, que poderia prejudicar o fluxo de validação cadastral.
- **Baixa precisão dos dados retornados**, exigindo reformulação da abordagem para garantir a assertividade na identificação de fraudes.
- **Desafios na arquitetura** para permitir a **migração do modelo on-premisse para cloud computing (AWS)**, garantindo maior escalabilidade.
- **Criação de um WebHook** para garantir a comunicação **assíncrona e passiva** entre as plataformas.

Além disso, foi necessário criar uma **arquitetura de microserviços escaláveis na AWS** capaz de se comunicar de forma **assíncrona** com o ambiente **on-premisse da CIRION**, onde utilizamos **RabbitMQ** para mensageria. Como na AWS o barramento de eventos é baseado no **Kafka**, desenvolvemos um **adapter/conector KYC**, garantindo **resiliência e entrega confiável entre os dois ambientes**.

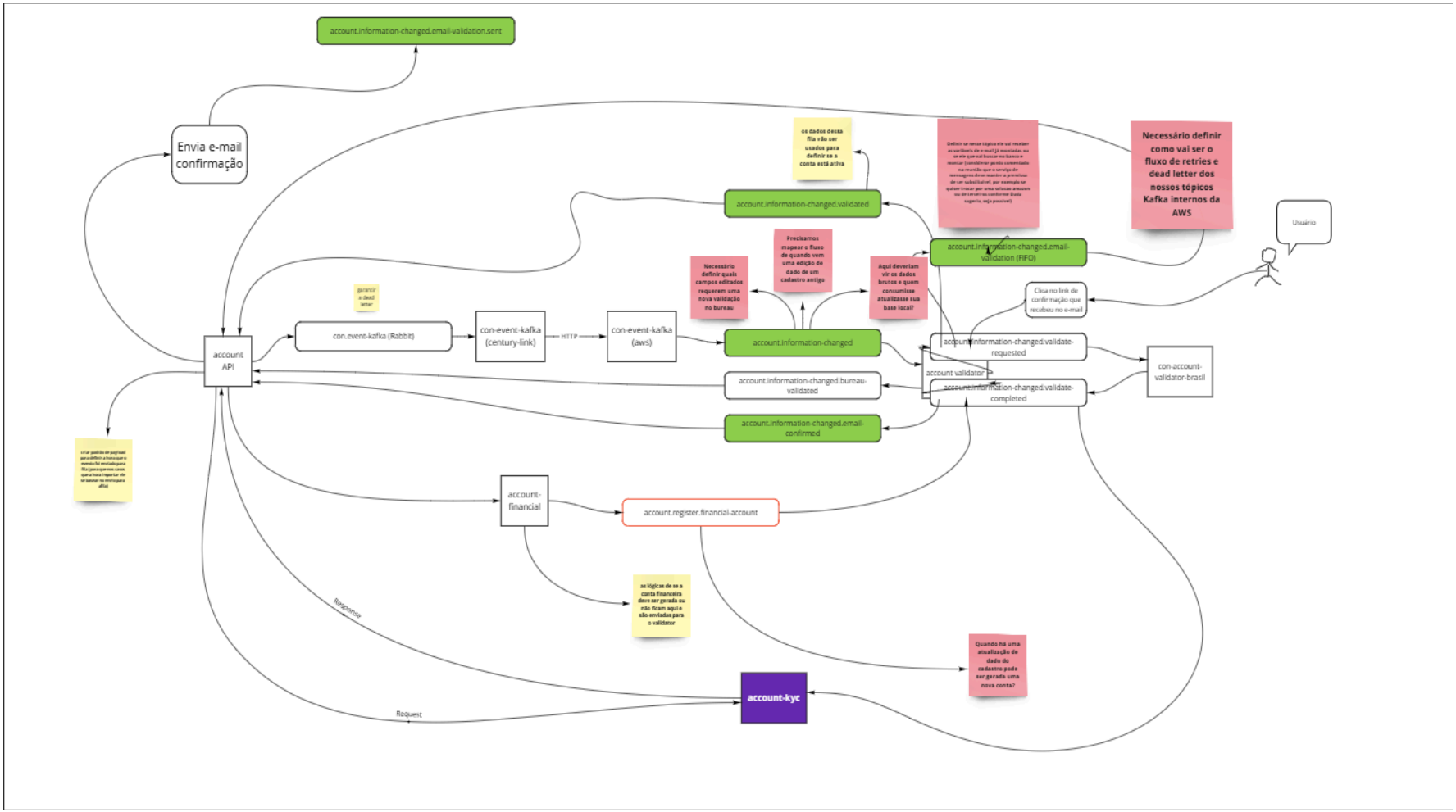
Diante desses desafios, a equipe aplicou esforços para:

1. **Criar um endpoint REST** para comunicação entre a plataforma Superbid e a IdWall, garantindo **processamento assíncrono e contínuo** dos dados dos usuários.
2. **Desenvolver uma nova arquitetura de microsserviços em AWS**, utilizando **Java, Spring Boot e Docker**, permitindo **escalabilidade automática** do sistema.
3. **Implementar um "Secret Manager"** para **filtragem e proteção de dados sensíveis**, garantindo **conformidade com a LGPD**.

4. **Refatorar os algoritmos de integração**, permitindo uma análise mais eficiente e segura dos dados coletados.
5. **Criar um mecanismo de adaptação entre RabbitMQ (on-premisse) e Kafka (AWS)**, garantindo **entrega confiável e sincronização eficiente de eventos**.

Elemento Tecnológico Inovador

1. **Inovação na verificação de identidade**: Antes do projeto, a **validação de identidade era feita manualmente** e não havia um sistema robusto de **background check**. Com a integração da IdWall, foi possível:
    - Criar um **mecanismo automático de avaliação de risco**.
    - Ampliar a **capacidade de análise de usuários** em tempo real.
    - Estabelecer uma **integração assíncrona via WebHook**, algo inexistente antes.
2. **Migração para Arquitetura Cloud**: A transição de um modelo **on-premisse para cloud** possibilitou:
    - **Redução de custos operacionais**.
    - **Aumento da capacidade de armazenamento e processamento** de dados cadastrais.
    - **Maior segurança e redundância dos dados**.
3. **Adaptação de modelos para inteligência artificial futura**: Com a nova estrutura, foi possível preparar a plataforma para:
    - **Coleta e estruturação de dados históricos**, permitindo futuras aplicações de **machine learning** para detectar padrões de fraude.
    - **Automação do fluxo de compliance**, reduzindo a necessidade de intervenção humana.
4. **Desenvolvimento do Conector KYC**: Para viabilizar a comunicação eficiente entre RabbitMQ e Kafka, desenvolvemos um **adapter/conector KYC**, garantindo:
    - **Resiliência e tolerância a falhas na troca de mensagens entre os dois ambientes**.
    - **Garantia de entrega confiável de eventos críticos de validação cadastral**.
    - **Gerenciamento de filas e retries em caso de falha na entrega de mensagens**.


account.information-changed.email-validation.sent
Envia e-mail confirmação
os dados dessa fila vão ser usados para definir se a conta está ativa
Necessário definir como vai ser o fluxo de retries e dead letter dos nossos tópicos Kafka internos da AWS
account.information-changed.validated
Usuário
account.information-changed.email-validation (FIFO)
Precisamos mapear o fluxo de quando vem uma edição de dado de um cadastro antigo
Necessário definir quais campos editados requerem uma nova validação no bureau
Aqui deveriam vir os dados brutos e quem consumisse atualizasse sua base local?
Clica no link de confirmação que recebeu no e-mail
garantir a dead letter
account API
con.event-kafka (Rabbit)
con-event-kafka (century-link)
HTTP
con-event-kafka (aws)
account.information-changed
account.information-changed.validate-requested
account validator
con-account-validator-brasil
account.information-changed.bureau-validated
account.information-changed.validate-completed
account.information-changed.email-confirmed
account-financial
account.register.financial-account
as lógicas de se a conta financeira deve ser gerada ou não ficam aqui e são enviadas para o validator
Quando há uma atualização de dado do cadastro pode ser gerada uma nova conta?
Response
Request
account-kyc

Versão Revisada (para revalidação do MCTI)

Foi removido o nome do fornecedor uma vez que não queremos evidenciar isso e sim a arquitetura interna criada e riscos e inovações inerentes a P&D, sendo assim o título também foi alterado.

III. Malha de Serviços KYC para Arquitetura de Validação de Identidade

Os desenvolvimentos tecnológicos experimentais foram orientados à **construção de uma nova arquitetura de microserviços escaláveis** para suportar processos de **análise e validação cadastral de usuários**. O objetivo foi garantir **resiliência e interoperabilidade entre ambientes distintos**, permitindo a **comunicação segura e eficiente** entre a infraestrutura **on-premise (CIRION, utilizando RabbitMQ) e AWS (utilizando Kafka)**.

Além da comunicação entre os barramentos de eventos, o projeto foi estruturado dentro de uma **malha de serviços híbrida**, integrando **microsserviços especializados na AWS** para validar identidade e processar eventos de maneira **event-driven**. Essa abordagem permitiu que diferentes serviços especializados fossem ativados conforme a necessidade, otimizando o processamento e garantindo melhor escalabilidade.

Antes do projeto:

- O sistema de validação cadastral dependia de processos fragmentados, tornando a **análise de identidade ineficiente e suscetível a falhas**.
- Havia **dificuldade na integração entre diferentes ambientes de infraestrutura**, impactando a **resiliência da comunicação**.
- O fluxo de mensageria não possuía um mecanismo de **garantia de entrega e controle de falhas**.

Riscos Tecnológicos e Incertezas Superadas

Devido à **natureza experimental do projeto**, os desafios enfrentados incluíam:

- **Integração de duas infraestruturas distintas**, exigindo a criação de um **conector KYC** para garantir a comunicação confiável entre RabbitMQ e Kafka.
- **Resiliência da arquitetura**: necessidade de manter **garantia de entrega** e evitar perda de mensagens durante a conversação entre ambientes.
- **Gerenciamento de filas e retries**: foi essencial desenvolver uma estratégia para lidar com **falhas de conexão e latência**, garantindo a entrega dos eventos críticos.
- **Criação de um fluxo assíncrono para troca de eventos**, mantendo **compatibilidade entre diferentes tecnologias de mensageria**.
- **Implementação de uma malha de serviços híbrida** distribuída entre CIRION e AWS, suportando microsserviços especializados na **validação de identidade**.

Diante desses desafios, a equipe aplicou esforços para:

1. **Criar uma arquitetura base para comunicação entre RabbitMQ (CIRION) e Kafka (AWS)**, garantindo escalabilidade e confiabilidade.

2. **Desenvolver um conector KYC especializado**, responsável pela **orquestração e controle de fluxo de mensagens entre ambientes distintos**.
3. **Implementar um modelo de gerenciamento de retries e dead letter queue (DLQ)**, garantindo a **integridade dos eventos** mesmo diante de falhas na comunicação.
4. **Validar a prova de conceito (POC) da arquitetura**, permitindo a evolução futura para um modelo completo de análise de identidade com machine learning.
5. **Criar uma arquitetura event-driven na AWS**, com microsserviços validadores que processam eventos em tempo real, melhorando a eficiência e escalabilidade do sistema.

Elemento Tecnológico Inovador

1. **Desenvolvimento de Arquitetura de Mensageria Resiliente**:
    - Integração entre **RabbitMQ (on-premisse)** e **Kafka (AWS)**, garantindo a **entrega confiável de eventos**.
    - Mecanismo de **gerenciamento de filas** e **reprocessamento automatizado** para evitar perda de dados.
2. **Conector KYC como Mecanismo de Interoperabilidade**:
    - Desenvolvimento de um **módulo de adaptação** que permite **a comunicação fluida entre dois sistemas de mensageria distintos**.
    - Camada intermediária para **normalização de eventos**, garantindo **compatibilidade entre tecnologias**.
3. **Prova de Conceito para Arquitetura Escalável**:
    - A fase inicial do projeto focou na **validação da comunicação assíncrona**, permitindo que a equipe testasse a confiabilidade da solução antes da implementação final.
    - Os esforços de P&D resultaram em um modelo que **não existia anteriormente** dentro da plataforma e que poderá ser expandido para futuras integrações com modelos analíticos mais complexos.
4. **Arquitetura Event-Driven em Ambiente Híbrido**:
    - Introdução de **microsserviços especializados na AWS** que operam em modelo event-driven para validação de identidade.
    - Processamento de eventos em tempo real, garantindo maior **eficiência, escalabilidade e otimização da carga do sistema**.
    - Independência operacional entre os serviços de validação, permitindo maior flexibilidade para adaptações futuras.