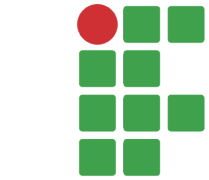


Instituto Federal de Educação, Ciência e Tecnologia da Paraíba

Campus João Pessoa

Unidade Acadêmica de Informação e Comunicação

TÍTULO DO TRABALHO

NOME DO DICENTE

JOÃO PESSOA

2022

Instituto Federal de Educação, Ciência e Tecnologia da Paraíba

Campus João Pessoa

Unidade Acadêmica de Informação e Comunicação

Título do trabalho

Nome do dicente

Relatório de Estágio Supervisionado apresentado à unidade curricular de Estágio Obrigatório do Curso Superior de Tecnologia em Redes de Computadores do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, como requisito parcial para obtenção do grau de Tecnólogo em Redes de Computadores.

Orientador: Prof. Dr. Paulo Ditarso Maciel Jr.

Supervisor: Nome do Supervisor

Coordenador do Curso: Prof. Me. José Gomes Quaresma Filho

Empresa: Nome da Empresa

Período: dia/mês/ano a dia/mês/ano

APROVAÇÃO

Prof. Dr. Paulo Ditarso Maciel Jr.
Orientador

Prof. Dr. Fulano de Tal
Avaliador

Prof. Me. Ciclano de Tal
Avaliador

Beltrano de Tal
Supervisor da empresa

Prof. Me. José Gomes Quaresma Filho
Coordenador do Curso Superior de Tecnologia
em Redes de Computadores

Nome do Discente
Estagiário

Aprovado em DIA de MÊS de ANO.

Visto e permitida a impressão
João Pessoa

(exemplo...) Este trabalho é dedicado às crianças adultas que, quando pequenas, sonharam em se tornar cientistas.

AGRADECIMENTOS

Dedico ...

RESUMO

Durante o estágio no ...

Palavras-chaves: 1a palavra-chave, 2a palavra-chave, 3a palavra-chave, 4a palavra-chave.

ABSTRACT

During the internship ...

Key-words: 1st keyword, 2nd keyword, 3rd keyword, 4th keyword.

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

AWS	Amazon Web Services
EC2	Elastic Compute Cloud
IFPB	Instituto Federal da Paraíba
...	

SUMÁRIO

1 INTRODUÇÃO

O presente relatório descreve as atividades de desenvolvimento Web back-end realizadas durante o trabalho no projeto Capana executado na Korporate no setor de fábrica de software, com carga horária semanal de 40 horas. O período de vigência do estágio aluno trabalhador foi de um de fevereiro de 2024 a 30 de abril de 2024.

Antes da implementação do projeto Capana, o ERP do contrante do projeto oferecia funcionalidades como a elaboração de propostas para aquisição de cartões e a atualização dos dados dos solicitantes dessas propostas. Esses processos incluíam a comunicação com birôs externos para a obtenção do score de inadimplência dos solicitantes e a integração com IDTechs para a verificação de identidade. Tal abordagem implicava em uma elevada complexidade na arquitetura do sistema, além de dificultar a sua manutenção.

Na qualidade de analista de sistemas, integrei a equipe encarregada de desenvolver o projeto capana. Fui incumbido de analisar, desenvolver e realizar testes manuais nos microsserviços desenvolvidos durante o projeto.

Essa análise envolveu a entrega de desenhos arquiteturais das soluções e diagramas de atividades, bem como a responsabilidade de desenvolver os módulos, implementando documentações no código, aderindo às boas práticas de programação e às diretrizes de codificação estabelecidas pela empresa, e ser responsável por definir e executar testes manuais funcionais e não funcionais no ambiente de homologação do cliente, a fim de garantir que o sistema atenda aos requisitos e expectativas.

1.1 Objetivos

O projeto capana, tem como objetivo, elaboração de propostas para aquisição de cartões e a atualização dos dados dos solicitantes dessas propostas no ERP do contratante do projeto, buscando simplificar a arquitetura do sistema. Para cumprir o objetivo geral, foi definido os seguintes objetivos específicos:

- Desenhos arquiteturais dos serviços entregues durante o projeto, como também diagramas de fluxo.
- Desenvolvimento de microsserviços para separar responsabilidades.
- Especificação e execução dos testes funcionais e não funcionais.

1.2 Korporate

A Korporate é uma empresa de alcance nacional, contando com sete anos de atividades, destacando-se pelos seguintes produtos principais: ERP para pequenas e médias empresas do varejo, o Pix Multibancos, um software que gerencia pagamentos via Pix e facilita a criação de Pix, o TakePay, software responsável por conceder cashback aos consumidores que atendam a determinados requisitos, e o Orquestrador, software encarregado de executar tarefas em lote com base nas ações fornecidas. Além de seus produtos internos, a Korporate mantém um setor de fábrica de software, dedicado ao desenvolvimento de soluções personalizadas conforme as necessidades de seus clientes.

1.3 Estrutura do Relatório

Este relatório está estruturado em cinco capítulos interligados:

- **Introdução**: Apresenta o contexto geral do estágio, delimitando o escopo do projeto e seus objetivos.
- **Fundamentação Teórica**: Fornece o embasamento teórico necessário para a compreensão das atividades realizadas, destacando as principais tecnologias e conceitos utilizados.
- **Descrição das atividades realizadas**: Detalha os serviços desenvolvidos durante o projeto.
- **Resultados Obtidos**: Apresenta os resultados obtidos após a entrega do projeto.
- **Considerações Finais**: Apresenta as principais conclusões obtidas durante o projeto, avaliando a experiência e sua contribuição para a formação profissional.

2 FUNDAMENTAÇÃO TEÓRICA

Nesse capítulo será demonstrada uma visão geral sobre as principais tecnologias utilizadas e a metodologia de gerenciamento de projeto utilizada.

2.1 UML

A UML (Unified Modeling Language) é uma linguagem gráfica que facilita a criação de softwares. Ela permite visualizar e detalhar os componentes de um sistema, como classes, objetos e suas interações, além de especificar o comportamento do sistema e documentar suas características. Tudo isso de forma padronizada e com o uso de diagramas, o que torna a comunicação entre os desenvolvedores mais fácil e eficiente (MIRO, 2024). A referida tecnologia foi empregada para o desenho do fluxo dos microsserviços e da arquitetura, sendo escolhida por ser uma ferramenta de mercado para o desenho de arquiteturas de software e fluxos da aplicação.

2.2 SQL

A Linguagem de Consulta Estruturada (Structured Query Language - SQL) se configura como uma linguagem de programação padronizada para gerenciamento de dados em bancos de dados relacionais. Sua função primordial reside na manipulação, organização e recuperação de informações armazenadas nesses bancos, possibilitando aos usuários a consulta, inserção, atualização e exclusão de dados (AWS, 2024).

A mencionada tecnologia foi utilizada na elaboração de *procedures* no banco de dados do cliente e na criação das tabelas necessárias para os microsserviços, sendo selecionada em virtude dos bancos de dados serem relacionais.

2.3 SpringBoot

O Spring Boot é um framework de software livre e open-source baseado na plataforma Java, projetado para facilitar e agilizar o desenvolvimento de aplicações web robustas e escaláveis. Sua principal característica reside na autoconfiguração, dispensando a necessidade de configurações manuais complexas e simplificando drasticamente o processo de criação de aplicações. Com o spring é possível criar aplicações Rest, Cloud, microsserviços entre outras arquiteturas (VMWARE, 2024).

A tecnologia em questão foi empregada no desenvolvimento dos microsserviços que compõem o projeto, mais especificamente: broker, onboarding, rotina de inativação de cli-

ente e cadastro ou atualização de cliente. Tal escolha tecnológica decorreu da experiência dos profissionais seniores e plenos.

2.4 SpringBootWeb

Spring Web é um componente do Spring Framework que fornece suporte para a criação de aplicações web, incluindo recursos para desenvolvimento de controladores, gerenciamento de solicitações HTTP, manipulação de sessões e cookies, entre outros (SANTANA, 2016). Este componente foi empregado nos microsserviços para disponibilizar endpoints de comunicação via HTTP, sendo escolhido por ser um componente padrão dentro do ecossistema Spring.

2.5 SpringWebFlux

Spring WebFlux é um framework dentro do ecossistema Spring, desenvolvido para facilitar a construção de aplicativos reativos em Java. Ele oferece uma abordagem baseada em programação reativa para lidar com solicitações HTTP e interações assíncronas, permitindo que os desenvolvedores criem aplicativos altamente escaláveis e eficientes em termos de recursos (BRITO, 2019). Este framework foi empregado nos seguintes microsserviços: cadastro ou atualização de cliente, rotina de inativação de cliente e onboarding, visando efetuar requisições HTTP para o broker. A justificativa para sua escolha baseou-se na gestão mais avançada de requisições e na capacidade de realizar solicitações assíncronas.

2.6 Flyway

Flyway é uma ferramenta open-source específica para o ecossistema Java, que visa gerenciar versões dos bancos de dados relacionais a partir de arquivos SQL. A ferramenta exige que os arquivos sigam um padrão de nomeação, indicando o número da versão do script e o seu propósito (VITOR, 2024). Essa ferramenta foi utilizada nos microsserviços que requerem um banco de dados, para o controle das migrações do banco de dados em ambientes de homologação e produção.

2.7 SWT

A tecnologia SWT (Standard Widget Toolkit) é uma biblioteca gráfica destinada ao desenvolvimento de interfaces de usuário em Java. Desenvolvida pela Eclipse Foundation, é usada principalmente no ambiente de desenvolvimento Eclipse, mas pode ser empregada em outras aplicações java (KESTERMANN, 2020). A referida tecnologia foi utilizada para o desenvolvimento do ERP do contratante do projeto, pois era a única tecnologia disponível, no ano 2000, para a construção de aplicações desktop multiplataforma em Java.

2.8 Metodologia

O Kanban, que em japonês significa "cartão visual", é um método de gestão de fluxo de trabalho que visa otimizar a entrega de valor através da visualização, limitação do trabalho em andamento e foco na melhoria contínua. Ele surgiu na Toyota na década de 1950 e se consolidou como uma das principais metodologias ágeis, utilizado em diversos contextos, desde o desenvolvimento de software até o gerenciamento de projetos e atividades pessoais, possuindo os seguintes princípios (SABINO, 2023):

1. **Visualizar o Fluxo de Trabalho:** O Kanban se destaca por sua natureza visual, utilizando um quadro físico ou digital para representar as etapas do processo e o *status* das tarefas. Essa representação gráfica facilita a compreensão do fluxo de trabalho, a identificação de gargalos e a comunicação entre os membros da equipe.

2. **Limitar o Trabalho em Andamento (WIP):** O Kanban estabelece limites para o número de tarefas que podem estar em cada etapa do processo, impedindo o acúmulo excessivo de trabalho e promovendo o foco em atividades prioritárias. Essa limitação ajuda a reduzir o desperdício, aumentar a produtividade e garantir um fluxo de trabalho mais fluido.

3. **Enfatizar a Entrega Contínua:** O Kanban incentiva a entrega frequente de valor ao cliente, seja mediante funcionalidades completas ou incrementos menores. Essa abordagem permite que os feedbacks sejam coletados e incorporados rapidamente, ajustando o curso do projeto conforme as necessidades do cliente.

4. **Tornar Políticas Explícitas:** As regras e políticas que governam o fluxo de trabalho no Kanban são tornadas explícitas e visíveis para todos os envolvidos. Isso garante a transparência do processo, facilita a resolução de conflitos e promove a padronização das atividades.

5. **Melhorar continuamente:** O Kanban incentiva a cultura da melhoria contínua, através da identificação e eliminação de desperdícios, otimização do fluxo de trabalho e experimentação de novas ideias. Essa mentalidade proativa leva a um processo em constante evolução e adaptação às necessidades do ambiente.

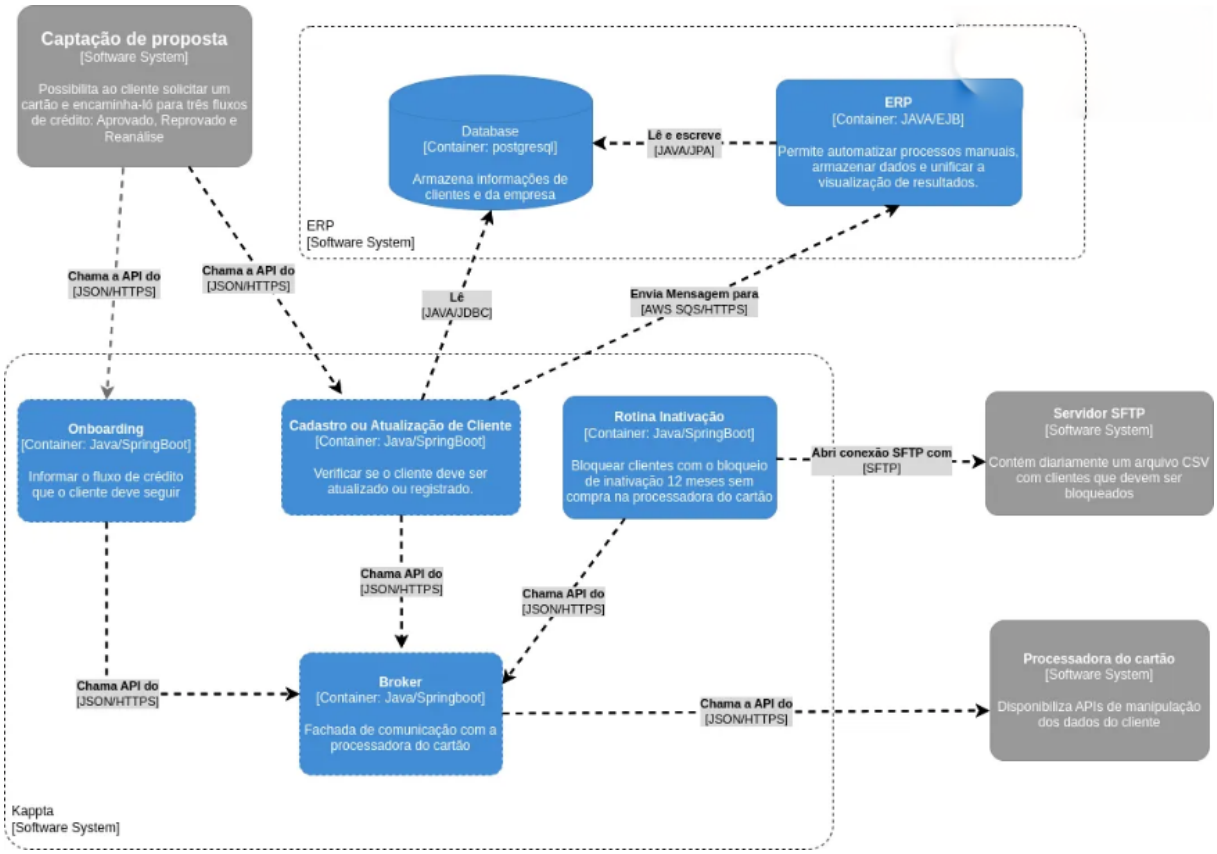
Esse método foi selecionado para a gestão do projeto devido à sua capacidade de limitar o fluxo de trabalho em andamento, priorizar tarefas que geram valor para o cliente e proporcionar uma visualização clara do fluxo de trabalho tanto para os membros do projeto quanto para o cliente.

3 DESCRIÇÃO DAS ATIVIDADES REALIZADAS

Este projeto tem como objetivo remover as funcionalidades de elaboração de propostas para aquisição de cartões e a atualização dos dados dos solicitantes dessas propostas no ERP do contratante do projeto, buscando simplificar a arquitetura do sistema. Atualmente, essa funcionalidade inclui integrações com birôs externos para verificar se o score de inadimplência do cliente está dentro do intervalo tolerado, além de uma integração com uma IDTech para validar a identificação do cliente, que envolve a captura de selfie e dos documentos de identificação, bem como a sincronização da base de dados de clientes com a processadora do cartão.

Para atender a essas necessidades e simplificar a arquitetura do sistema, foi elaborada uma solução baseada em microsserviços. A Figura 1 ilustra a arquitetura proposta, onde cada microsserviço é responsável por uma parte do processo de onboarding, garantindo maior escalabilidade e facilidade de manutenção. Nas subseções seguintes, cada um desses microsserviços será detalhado, explicando sua função e como interagem entre si.

Figura 1 – Arquitetura do projeto



Fonte: próprio autor

3.1 Broker

Microsserviço cuja finalidade é atuar como um broker, que é um componente centralizado que gerencia a comunicação entre diferentes partes do sistema (STEEN; TANENBAUM, 2024). Seguindo esse conceito, o broker funcionará como intermediador na comunicação entre os serviços internos e a processadora do cartão, que é um serviço externo. Esse microsserviço é essencial para o ecossistema, pois evita a duplicação de configurações de integração nos microsserviços que dependem dos dados da processadora de cartão.

3.2 Onboarding

O onboarding é um microsserviço que foi implementado utilizando SpringBootWeb aderindo ao padrão MVC, disponibilizando uma API que visa informar o fluxo de crédito que o cliente deve seguir quando solicita um cartão, que são respectivamente:

1. **Nova proposta**: Quando o cliente não possui inadimplências, carnê ou cheque e não possui o cartão informado.
2. **Reanálise de crédito**: Quando o cliente possui o cartão, porém está bloqueado por inatividade de compras por 12 meses.
3. **Reprovado**: caso possua inadimplências, carnê ou cheque, ou possua o cartão ativo, ou possua o cartão bloqueado, porém o tipo de bloqueio diferente de inativação.

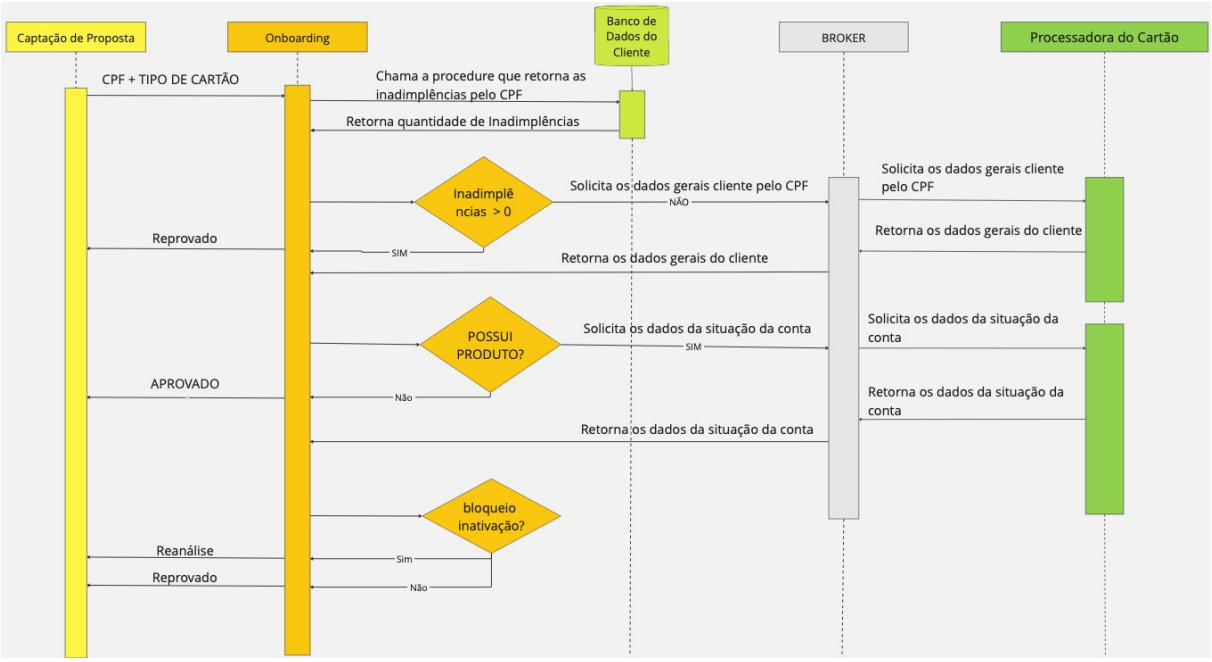
A Figura 2 ilustra o diagrama de fluxo do microsserviço de onboarding de clientes, detalhando as quatro etapas principais desse processo: captação de proposta, verificação de inadimplências, consulta à processadora de cartão e tomada de decisão final. A seguir, cada uma dessas etapas será descrita em detalhes, evidenciando a lógica e as interações entre os componentes do sistema.

Após o solicitante da proposta informar o **cpf** e o **cartão desejado** no **sistema de captação de proposta**, essas informações são encaminhadas ao microsserviço de **onboarding**. Este, por sua vez, consulta o banco de dados do contrante do projeto para verificar a existência de pendências financeiras a partir do cpf.

Caso sejam identificadas inadimplências, o **sistema de captação de proposta** é notificado para reprovar a proposta. Se o cliente estiver em dia com suas obrigações, o **onboarding** realiza uma consulta ao broker utilizando o **cpf** do cliente.

O **broker**, atuando como intermediário, encaminha essa solicitação à **processadora de cartão**, que retorna as informações cadastrais do cliente. Com os dados obtidos, o **onboarding** verifica se o cliente já possui o cartão solicitado.

Figura 2 – fluxo do onboarding



Fonte: próprio autor

Caso o cliente ainda não possua o cartão, o **sistema de captação de proposta** é notificado para seguir o fluxo de nova proposta. No entanto, se o cliente já possui o cartão, o **onboarding** verifica o status da conta. Se a conta estiver inativa o **sistema de captação de proposta** é notificado para seguir o fluxo de reanálise de crédito. Caso contrário, **sistema de captação de proposta** é notificado para reprovar a proposta.

O microsserviço de onboarding desempenha um papel crucial na gestão de riscos e na otimização da carteira de clientes. Ao impedir a concessão de crédito a indivíduos com histórico de inadimplência, a empresa minimiza o risco de inadimplências futuras, protegendo seu patrimônio. Além disso, o sistema incentiva a aquisição do cartão de crédito pela loja, desestimulando o uso de outras formas de pagamento, como o carnê, o que contribui para o aumento da rentabilidade e da fidelização dos clientes.

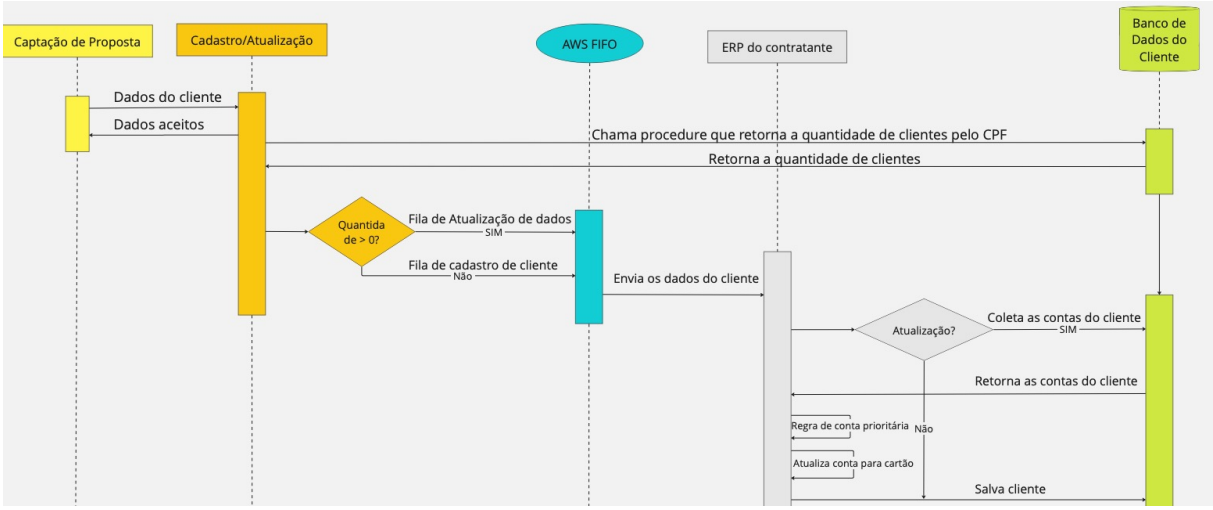
3.3 Cadastro/Atualização de cliente

O cadastro/atualização de cliente é um microsserviço, que foi implementado utilizando framework SpringBootWeb e seguindo a arquitetura MVC, disponibilizando uma API responsável por processar as propostas aprovadas e encaminhar os dados do cliente para um processo de cadastro de nova conta ou atualização dos dados na base de dados do contrante do projeto.

A Figura 3 ilustra o diagrama de fluxo do microsserviço de cadastro/atualização de cliente, detalhando as três etapas principais desse processo: recepção de propostas aprovadas,

validação e atualização dos dados do cliente e, por fim, o armazenamento das informações no banco de dados. A seguir, cada uma dessas etapas será descrita em detalhes, evidenciando a lógica e as interações entre os componentes do sistema.

Figura 3 – fluxo do cadastro/atualização de cliente



Fonte: próprio autor

Após a avaliação do perfil de crédito pelo microsserviço de onboarding, o cliente é direcionado para um dos seguintes fluxos de proposta: nova proposta ou reanálise de crédito. Para a aprovação da proposta, é necessário passar por duas etapas:

1. **Verificação de crédito**: É realizada uma consulta aos birôs de crédito para avaliar o score de inadimplência do cliente. Caso o score esteja dentro do limite permitido, o cliente avança para a próxima etapa.

2. **Atualização cadastral e validação biométrica**: O cliente atualiza seus dados cadastrais e realiza a captura de uma selfie para fins de validação biométrica caso estiver no fluxo de nova proposta. Após essa etapa as informações preenchidas são submetidas para o sistema de reconhecimento facial e documental (IDTech). Caso sistema identifique alguma inconsistência, a proposta é reprovada.

Após a aprovação, os dados da proposta são enviados ao microsserviço de cadastro/atualização. O microsserviço consulta a base de dados para verificar a existência de um cadastro associado ao cliente. Se o cadastro for encontrado, os dados da proposta são encaminhadas para a fila de atualização. Caso contrário, os dados da proposta é enviada para a fila de cadastro.

O ERP, ao receber a mensagem da fila, processa a solicitação, realizando o cadastro completo do cliente ou a atualização dos dados existentes, vinculando à conta ao tipo de cadastro fatura. Ao atualizar os dados de um cliente com múltiplas contas, o sistema escolhe a conta com

o tipo de cadastro que deve ser priorizada para o processo de atualização descartando as outras na seguinte ordem: fatura, carnê e por último cheque ou cartão de crédito/vista(não pertence ao contrante do projeto).

O microsserviço de cadastro/atualização desempenha um papel crucial na manutenção da integridade dos dados do cliente, evitando a duplicidade de contas e indicando ao ERP qual a ação a ser tomada: cadastro ou atualização. Essa integração com o ERP permite a migração de contas de outros tipos de pagamento (carnê, cheque, cartão de crédito/vista) para o cartão próprio do contratante do projeto.

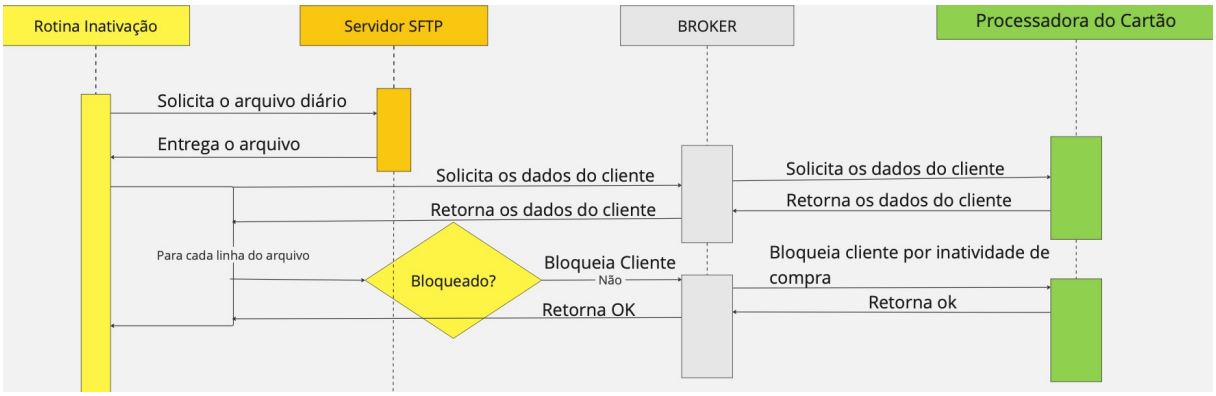
Essa migração oferece benefícios ao contratante do projeto, como: maior comodidade na gestão de pagamentos, disponibilizar benefícios exclusivos ao uso do cartão, ofertar programas de pontuação e condições especiais, fortalecendo assim o vínculo com a empresa e aumentando a probabilidade de novas aquisições. Além de **transferir** a responsabilidade da gestão dos dados do contrante do projeto para a processadora do cartão.

3.4 Rotina de inativação de cliente

A rotina de inativação é um microsserviço que foi implementado utilizando o framework SpringBootWeb aderindo ao padrão MVC, que tem como objetivo bloquear clientes que estão a 12 meses sem comprar no cartão de crédito.

A Figura 4 ilustra o diagrama de fluxo do microsserviço de rotina de inativação, detalhando as três etapas principais desse processo: download de arquivo com clientes que devem ser bloqueados, verificação dos status da conta na processadora do cartão e, por fim, o bloqueio do cliente. A seguir, cada uma dessas etapas será descrita em detalhes, evidenciando a lógica e as interações entre os componentes do sistema.

Figura 4 – fluxo da rotina de inativação



Fonte: próprio autor

O microsserviço de rotina de inativação disponibiliza um job diário agendado às 00:00 a fim de baixar um arquivo CSV com uma lista de clientes que devem ser bloqueados com o tipo

de bloqueio de inativação há mais de 12 meses sem compras no servidor SFTP.

Para cada cliente na lista, o job consulta o broker para obter as informações sobre a conta do cliente que por sua vez encaminha para a processadora do cartão. Caso a conta esteja ativa, o job envia uma solicitação ao broker para bloquear a conta do cliente, que por sua vez encaminha para a processadora do cartão.

O microsserviço de rotina de inativação desempenha um papel importante para a manutenção do processo de reanálise de crédito no sistema de captação de proposta, pois esse processo só ocorre em clientes com bloqueio de inativação de 12 meses. Permitindo também identificar clientes inativos, a fim de analisar o seu comportamento, possibilitando a criação de programas de incentivo personalizados para estimular novas compras.

4 RESULTADOS OBTIDOS

Esta seção apresenta os resultados decorrentes da implementação do projeto Capana, os quais serão analisados sob duas perspectivas principais: o impacto nas operações das lojas e as alterações no sistema ERP.

4.1 Impacto nas Operações das Lojas

- **Otimização do Fluxo de Caixa**: A transferência do processo de obtenção e desbloqueio de cartões para dispositivos móveis (celulares e tablets) alocados nas lojas resultou em uma significativa melhoria na fluidez dos caixas. A eliminação da necessidade de executar essas operações nos caixas liberou os operadores para se concentrarem nas transações de venda, reduzindo o tempo de espera dos clientes e aumentando a taxa de conversão. Pode-se inferir que a redução da complexidade das operações de caixa contribui para um ambiente de atendimento mais eficiente e satisfatório para o consumidor.
- **Incremento na Captação de Propostas**: A implementação de um protocolo de questionamento aos clientes no momento do checkout, indagando sobre o interesse em adquirir ou desbloquear um cartão, demonstrou um aumento na quantidade de propostas.

4.2 Alterações no Sistema ERP

A migração da funcionalidade de elaboração de propostas para um ambiente escalável e desacoplado do ERP proporcionou benefícios em termos de arquitetura de sistema. A remoção dessa carga de processamento do ERP resultou em um sistema mais leve e responsivo, facilitando a manutenção e reduzindo o risco de impactos negativos em outras funcionalidades em caso de falhas ou necessidade de atualizações. A adoção de uma arquitetura desacoplada promove a modularidade e a resiliência do sistema, permitindo intervenções e melhorias específicas sem comprometer a estabilidade do conjunto.

5 CONSIDERAÇÕES FINAIS

O desenvolvimento deste projeto revestiu-se de suma importância para o contratante, ao promover a migração de funcionalidades complexas do ERP para uma arquitetura escalável de microsserviços. A independência entre os microsserviços facilita a manutenção e a implementação de novas regras de negócio, assegurando o cumprimento das metas estabelecidas no projeto.

Durante a execução do projeto, foi possível aplicar, na prática, os conhecimentos adquiridos ao longo do curso, tais como padrões de projeto, desenvolvimento de aplicações web com Spring Boot, consultas SQL e outros, proporcionando ao aluno uma valiosa experiência nessas áreas.

Ademais, o projeto propiciou ao aluno a compreensão do funcionamento do ecossistema de pagamentos com cartão de crédito pertencente ao contratante.

Como trabalho futuro, propõe-se aprofundar a investigação sobre a viabilidade e os benefícios da migração do sistema ERP legado do contratante para uma plataforma tecnológica mais moderna e alinhada com as tendências de mercado. Esta migração visa não apenas a atualização tecnológica, mas também a otimização de processos, o aumento da eficiência operacional e a melhoria da escalabilidade e manutenibilidade do sistema.

REFERÊNCIAS BIBLIOGRÁFICAS

AWS. *O que é SQL (linguagem de consulta estruturada)?* 2024. Acesso em: 31 de Julho de 2024. Disponível em: <https://aws.amazon.com/pt/what-is/sql/>. Citado na página 12.

BRITO, M. *Spring Webflux*. 2019. Acesso em 1 de Agosto de 2024. Disponível em: <https://medium.com/@michellibrito/spring-webflux-f611c8256c53>. Citado na página 13.

KESTERMANN, T. *The Standard Widget Toolkit (SWT) — Java UI at its Best (Part 1)*. 2020. Acesso em 1 de Agosto de 2024. Disponível em: <https://medium.com/@TorstenKestermann/the-standard-widget-toolkit-swt-java-ui-at-its-best-part-1-444f7f15b74>. Citado na página 13.

MIRO. *O que é um diagrama UML*. 2024. Acesso em: 13 de Abril de 2024. Disponível em: <https://miro.com/pt/diagrama/o-que-e-uml/>. Citado na página 12.

SABINO, R. *Kanban: o que é, o Método Kanban, principais conceitos e como funciona no dia a dia*. 2023. Acesso em: 22 de Outubro de 2024. Disponível em: <https://www.alura.com.br/artigos/metodo-kanban>. Citado na página 14.

SANTANA, E. *Desenvolvendo uma Aplicação Web com Spring Boot e Spring MVC*. 2016. Acesso em 1 de Agosto de 2024. Disponível em: <https://www.devmedia.com.br/desenvolvendo-uma-aplicacao-web-com-spring-boot-e-spring-mvc/34122>. Citado na página 13.

STEEN, M. van; TANENBAUM, A. S. *Distributed Systems*. 4rd. ed. [s.n.], 2024. E-book. Disponível em: <https://www.distributed-systems.net/index.php/books/ds4/>. Citado na página 16.

VITOR, O. *O que é Flyway e por que usa-lo? Com Java e Spring!* 2024. Acesso em 1 de Agosto de 2024. Disponível em: <https://medium.com/@perez_vitor/o-que-%C3%A9-flyway-e-por-que-usa-lo-com-java-e-spring-312219ebf840>. Citado na página 13.

VMWARE. *Why Spring?* 2024. Acesso em: 31 de Junho de 2024. Disponível em: <https://spring.io/why-spring>. Citado na página 12.