**Entendimento da Demanda**

**[WI-13477 Pagamentos Open Banking]**

**Histórico:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Versão** | **Data** | **Descrição** | **Responsável** |
| 01 | 06/02/2020 | Confecção do documento | Alessandro Ferreira da Silva |
| 01 | 05/06/2020 | Requisitos de negócio | Steven Ramos da Silva |

Sumário

[1. Narrativa de Negócio: 3](#_Toc42261192)

[2. As Is: 3](#_Toc42261193)

[3. Repositórios 4](#_Toc42261194)

[4. To Be: 5](#_Toc42261195)

[5. Requisitos de negócio 7](#_Toc42261196)

[6. Solução técnica: 7](#_Toc42261197)

[7. Narrativa Técnica: 9](#_Toc42261198)

[8. C4 Model 10](#_Toc42261199)

[a. Diagrama de Contexto 10](#_Toc42261200)

[b. Diagrama de Containers 11](#_Toc42261201)

[c. Diagrama de Componentes 12](#_Toc42261202)

[9. Contratos API Canal – Pagamentos 13](#_Toc42261203)

[10. Referências 15](#_Toc42261204)

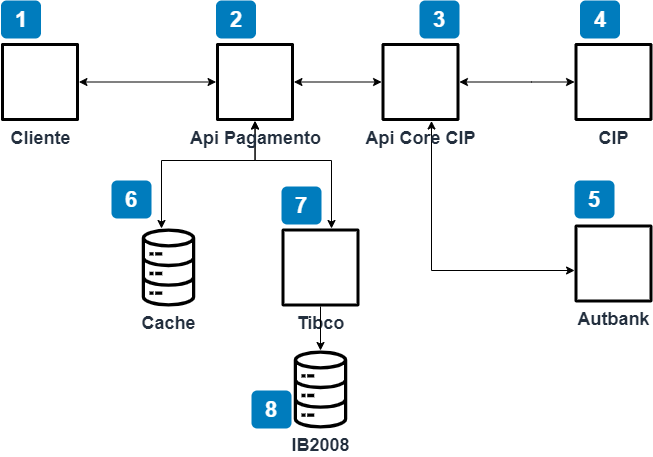
[11. Pré-Requisito 15](#_Toc42261205)

[12. Estimativa 15](#_Toc42261206)

# Narrativa de Negócio:

Eu **como** banco ABC Brasil, **preciso** que seja desenvolvido uma API **para** processamento de pagamento no Canal Open Banking utilizando o novo Core de Pagamento.

# As Is:



Atualmente o Open Banking possui uma versão em homologação da Api de Pagamento que foi concebida na estrututura do Internet Banking PJ atual.

A Api possui 4 endpoints, são eles:

* Consulta CIP - Get

Endpoint responsável por validar se um código de barras está registrado.

A CIP é uma associação civil sem fins lucrativos que integra o Sistema de Pagamentos Brasileiro (SPB). A interface entre a api de pagamento e a CIP é feito via Tibco através de uma api e toda consulta CIP via api de pagamentos é armazenado temporariamente no redis.

* Envio Pagamento - Post

Endpoint responsável por enviar um boleto ou tricon para o processamento de pagamento.

Possui callback de notificação para o cliente.

* Consulta Situação - Get
  + Endpoint responsável por consultar a situação do pagamento.
* Comprovante - Get
  + Endpoint resposável por consultar um comprovante de pagamento.

A inclusão de pagamentos via api são unitárias e enfileiradas via Tibco, este por sua vez registra os dados em uma base de dados SQL Server chamada IB2008. O cliente recebe um protocolo para acompanhar o fluxo de processamento em um segundo momento.

O processamento é feito por robôs desenvolvidos em .net console application e um portal de apoio ao Internet Banking centraliza as configurações necessárias para uso e controles do ciclo de vida de um pagamento.

Lentidão, fluxo de processamento lento e dead locks na base de dados foram insumos que levaram o time técnico + time de negócios decidirem por criar uma nova Api plugando no core de pagamentos que vem sendo desenvolvido pelo time de Internet Banking PJ.

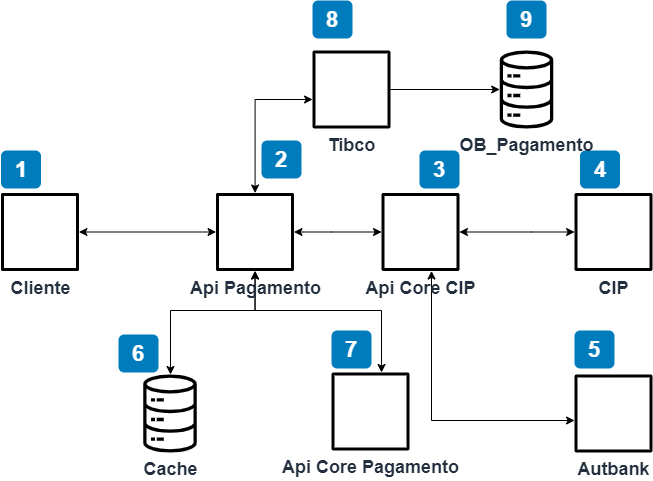
Estratégia essa adotada para que finalmente a Api de Pagamento do canal Open Banking seja oferecida como produto para as fintechs parceiras do banco ABC Brasil.

# Repositórios

Abaixo segue um mapeamento das referências de onde estão implementados cada método ilustrado anteriormente:

|  |  |
| --- | --- |
| **Repositórios** | |
| APi Pagamento | <https://abcbrasil.visualstudio.com/OpenBanking/_git/ABCBrasil.IB.Pagamento.Api> |
| WCF | <https://abcbrasil.visualstudio.com/InternetBanking/_git/IB-WCF> |
| ApoioIB | <https://abcbrasil.visualstudio.com/InternetBanking/_git/ApoioIB> |
| IB2008 | <https://abcbrasil.visualstudio.com/InternetBanking/_git/IB_MSTFS> |
| Robôs | PENDENTE |

# To Be:



Criar uma base de dados ABC\_OB\_PAGAMENTO para registrar informações do fluxo de pagamento de Open Banking.

Desenvolver uma api de canal pagamento que processa os pagamentos na nova Api Core de Pagamento.

Desenvolver um agendador para gerenciar lotes de processamento antes de enviar para a api core de pagamento.

Desenvolver um robô de callback responsável por notificar o cliente a cada etapa do processamento do pagamento.

Desenvolver e/ou integrar nos processos Tibco existentes referente ao fluxo de pagamento.

Desenvolver um Portal de apoio ao canal OpenBanking.

###Api Pagamento - Canal

validações gerais

se crc pertence ao login openbanking

se conta pertence ao crc informado

envio de pagamento

aprovação automática

idempotência

codigo de barras x identificador único do cliente

validações

geral

data e horarios por valores (atual base)

data de excessão (atual base)

data proximo dia util

se valor informado > 0

titulos

se possui titulo gravado no cache, caso contrario consultar na cip

se titulo apto

data limite de pagamento expirada

limite de pagamentos parciais atingido

se tipo de recebimento valor divergente (tpAutcRecbtVlrDivgte) permite o valor informado

tri.con (Ainda será plugado no novo core de processamento)

se possui o mesmo valor informado

se convenio é aceito

recuperar o favorecido do convenio (atual base)

consulta de situação

por protocolo

por identificador único do cliente

consulta de titulo na cip

armazenar em cache o resultado da cip para aplicar validações posteriores

calculo de titulo

por codigo de barras

callbacks (uso interno) - Callback do core

recebe alteração de situação

log de eventos

###Apoio Pagamentos - Canal

indicadores do canal

listagens

listar titulos processados/rejeitados

cancelar titulos pendentes de processamento

###Robo de Callback - Canal

notificar cliente quando ocorrer alteração de situação do pagamento

notificar cliente quando um pagamento for rejeitado no processo

###Tibco - Canal

incluir titulos na base do canal

alterar situação dos titulos da base do canal

###Compensação - Canal (ABBC)

compensar os titulos incluidos pelo canal

# Requisitos de negócio (Rascunho)

* 1. Precisamos garantir a idempotencia durante o pagamentos
  2. Possibilidade de efetuar pagamentos do tipo 1,2 e 3.
  3. Respeitar as regras de pagamento 1, 2 e 3 definidas pela CIP
  4. Todos os erros retornados ou mensagens de status devem estar em formato simples de leitura
  5. Os status do pagamento deveram retornar de forma correta em todas as consultas
  6. Armazenar os dados de pagamento nos Reds.
  7. Gerar comprovantes dos Pagamentos
  8. Garantir que os pagamentos sejam executados dentro das janelas de pagamento.
  9. Conseguir comportar uma grande quantidade de pagamentos podendo exceder mais de 30 mil.
  10. Permitir pagamentos de titulos que contém carência de juros, abatimento por pagamento antecipado e inclusão de juros ou mora.
  11. Respeitar tanto para desconto ou multa o tipo de calculo, podendo ser diário, mensal ou anual.
  12. Bloquear titulos que o pagamento após o vencimento não é permitido.
  13. Todas as movimentações deveram ser armazenadas na nova base de pagamentos.
  14. Todos os boletos deveram ser validados na cip antes que o processo aconteça
  15. O pagamento deverá ter persistência caso ocorra um problema durante o processamento ou envio.
  16. Todos os erros ou status deveram ser notificados imediatamente ao cliente.

# Solução técnica:

Abaixo a feature e suas respectivas user stories identificadas a serem desenvolvidas:

Feature Api Canal Pagamento

* 1. User Stories
     1. **Api Canal Pagamento - Consulta CIP**
        1. Criar um endpoint GET para consultar boleto na cip;
        2. Criar integração com a api de pagamentos do Tibco já existente;
        3. Salvar protocolos das consultas temporariamente no redis para consulta posterior na inclusão de pagamento;
     2. **Api Canal Pagamento - Inclusão de Pagamento**
        1. Criar endpoint POST para inclusão de pagamento unitário;
        2. Criar integração para cálculo de títulos no core CIP;
        3. Criar integração para cálculo de tributos e concessionária;
        4. Criar integração para inclusão de pagamento via api core de pagamento;
        5. Criar uma base de dados ABC\_OB\_PAGAMENTO;
        6. Criar tabelas para registro de informações inclusão de pagamento
        7. Gerar protocolo e notificar via Url de Callback o cliente.
     3. **Api Canal Pagamento - Consulta Pagamento**
        1. Criar endpoint GET para consulta de situação do pagamento incluído
        2. Consultar situação por protocolo
        3. Consultar situação por identificador único do cliente + protocolo
     4. **Api Canal Pagamento - Consulta Lista de Pagamentos**
        1. Criar endpoint GET para consulta de lista dos pagamento incluídos
        2. Lista paginada com valores de itens por página e range de data pré-determinados
        3. Possibilitar filtros por data, ordem, quantidade de itens por página
     5. **Api Canal Pagamento - Busca Comprovante**
        1. TBD
     6. **Api Canal Pagamento – Cálculo**
        1. TBD
     7. **Api Canal Pagamento – Cancelar Agendamento**
        1. TBD
     8. **Agendador Canal**
        1. TBD
     9. **Robo Canal de Callback**
        1. TBD
     10. **Tibco Canal Pagamento**
         1. TBD
     11. **Sas Canal Pagamento** 
         1. TBD

.

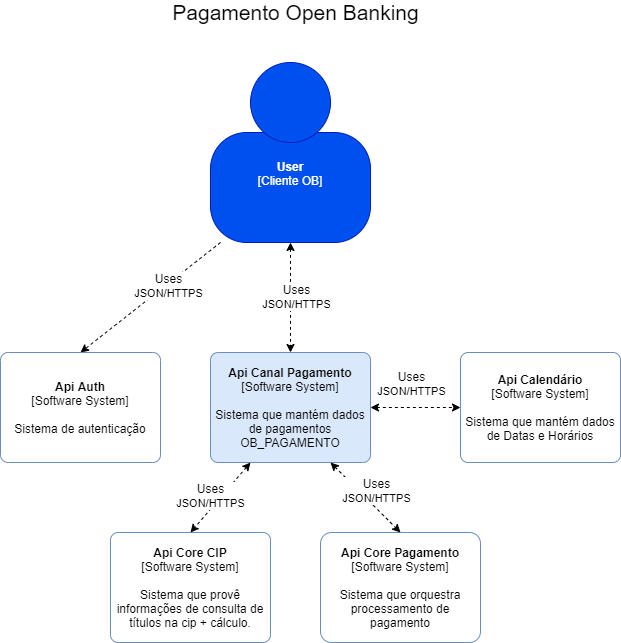
# Narrativa Técnica:

* Utilizar a api de autenticação atual para geração de Bearer Token;
* Utilizar CorrelationId para mapeamento do ciclo desde a requisição até a resposta devolvida para o cliente e garantir que as fronteiras utilizem este mesmo CorrelationId Axway, Tibco e Logs.
* Criar um repositório em ABCBrasil.OpenBanking.Pagamento.Api;
* Criar uma base de dados ABC\_OB\_PAGAMENTO;
* Aplicar registro eventos em todos enpoints da api com login, verbo, endpoint, payload request http statuscode e descrição de erro se existir;
* Aplicar validação via middleware para validar se código do cliente pertence ao login informado;
* Aplicar instrumentação de log via Seq;
* Aplicar Helth check;
* Aplicar Fast fail com Fluent Validation;
* Aplicar política de retry ao se comunicar com outra api para casos de erro HTTP 400;

# C4 Model

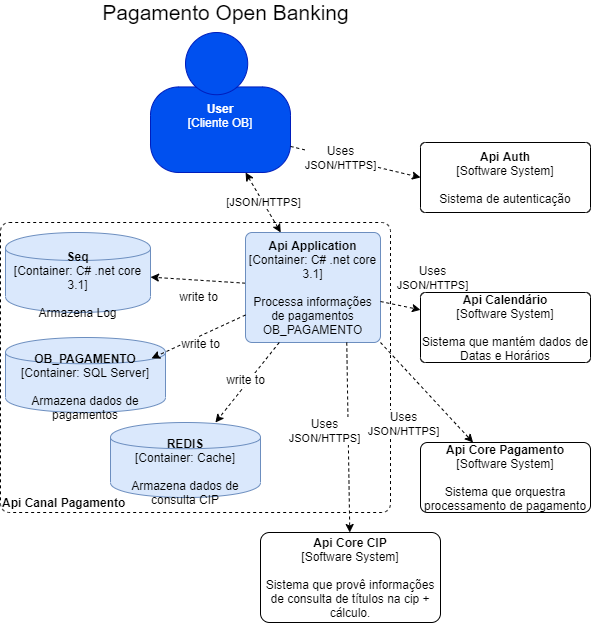
* 1. Diagrama de Contexto

O nível 1, um diagrama de contexto do sistema, mostra o sistema de software que você está construindo e como ele se encaixa no mundo em termos das pessoas que o utilizam e dos outros sistemas de software com os quais ele interage



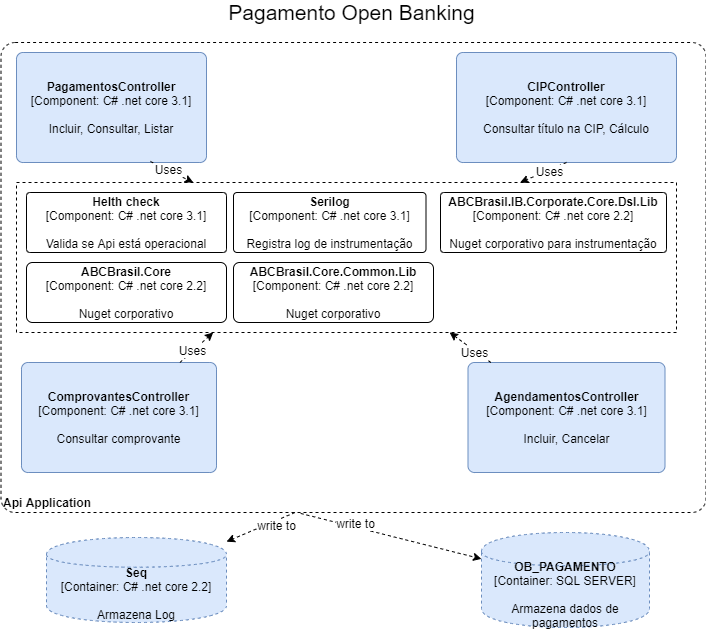
* 1. Diagrama de Containers

O nível 2, um diagrama de container, amplia o sistema de software e mostra os containers (aplicativos, armazenamentos de dados, microservices, etc.) que compõem esse sistema de software. As decisões de tecnologia também são uma parte fundamental desse diagrama.



* 1. Diagrama de Componentes

O nível 3, um diagrama de componentes, amplia um container individual para mostrar os componentes dentro dele. Esses componentes devem mapear para abstrações reais (por exemplo, um agrupamento de código) em sua base de código.



# Contratos API Canal – Pagamentos

|  |  |
| --- | --- |
| Id | 1 |
| Base de Dados | N/A |
| Objetivo | Consultar CIP |
| Tipo | Sincrono |
| Method | GET |
| Endpoint | api/v1/pagamentos/cip |
| Controller | CIPController.cs |
| Fast Fail | 1. Código do cliente obrigatório 2. Código de barras obrigatório |
| Regras de Negócio | 1. Validar Api Key 2. Validar Bearer Token 3. Consultar código de barras através da api Tibco pagamentos/boletos 4. Se localizado, incluir no cache e devolver sucesso com o payload response para o cliente. 5. Se não localizado, devolver um bad request com código 995 e a mensagem “Boleto não localizado na CIP” |
| Payload Request | api/v1/pagamentos/cip/{codigo do cliente}/{codigo de barras} |
| Status Response | 200 |
| Header | Authorization = {Bearer}  ABC\_KeyID\_Pagamento = {key}  X-CorrelationID = {valor} |
| Payload Response | Pendente link Swagger |

|  |  |
| --- | --- |
| Id | 2 |
| Base de Dados | ABC\_OB\_PAGAMENTO |
| Objetivo | Inclusão de Pagamento |
| Tipo | Assincrono |
| Method | POST |
| Endpoint | api/v1/pagamentos |
| Controller | PagamentosController.cs |
| Fast Fail | 1. Código do cliente obrigatório 2. Agência do cliente obrigatório 3. Conta do cliente obrigatório 4. Url de notificação obrigatório 5. Título obrigatório 6. Código do pagamento obrigatório 7. Data de pagamento se informado, igual ou superior a data atual 8. Data de pagamento se informado, validar data de exceção 9. Data de pagamento se informado, validar próximo dia útil 10. Data e horário de pagamento por valores 11. Valor do pagamento obrigatório superior a zero 12. Identificação com no máximo 25 carcteres |
| Regras de Negócio | 1. Validar Api Key 2. Validar Bearer Token 3. Validar se conta pertence ao cliente informado. 4. Validar se título já existe no cache, caso não exista, consultar código de barras através da api Tibco pagamentos/boletos. Se localizado, incluir no cache. 5. Se título e não localizado, devolver um bad request com código 995 e a mensagem “Boleto não localizado na CIP” 6. Validar se título e data limite de pagamento está expirado 7. Validar se título e limite de pagamentos parciais foi atingido 8. Validar se título e tipo (tpAutcRecbtVlrDivgte) permite valor divergente se caso informado 9. Validar se tricon possui o mesmo valor informado 10. Validar se convênio é aceito 11. Recuperar dados do favorecido do convênio 12. Incluir na base de dados com aprovação automática 13. Idempotência, ou seja, garantir que as requisições enviadas pelo cliente tenha sempre o mesmo resultado 14. Criar integração para cálculo de títulos na core de CIP 15. Criar integração para cálculo de tributos e concessionária 16. Criar integração para inclusão de pagamento via api core de pagamento |
| Payload Request | api/v1/pagamentos/cip/{codigoCrc}/{codigo de barras} |
| Status Response | 201 |
| Header | Authorization = {Bearer}  ABC\_KeyID\_Pagamento = {key}  X-CorrelationID = {valor} |
| Payload Response | Pendente link Swagger |

|  |  |
| --- | --- |
| Id | 3 |
| Base de Dados | ABC\_OB\_PAGAMENTO |
| Objetivo | Consultar Pagamento |
| Tipo | Sincrono |
| Method | GET |
| Endpoint | api/v1/pagamentos |
| Controller | PagamentosController.cs |
| Fast Fail | 1. Código do cliente obrigatório 2. Código de barras obrigatório |
| Regras de Negócio | 1. Validar Api Key 2. Validar Bearer Token 3. Consultar protocolo na base de dados ABCOBPJ |
| Payload Request | api/v1/pagamentos/{codigo do cliente}/{codigo de barras} |
| Status Response | 200 |
| Header | Authorization = {Bearer}  ABC\_KeyID\_Pagamento = {key}  X-CorrelationID = {valor} |
| Payload Response | Pendente link Swagger |

|  |  |
| --- | --- |
| Id | 4 |
| Base de Dados | ABC\_OB\_PAGAMENTO |
| Objetivo | Consultar Lista de Pagamentos |
| Tipo | Sincrono |
| Method | GET |
| Endpoint | api/v1/pagamentos |
| Controller | PagamentosController.cs |
| Fast Fail | 1. Código do cliente obrigatório 2. Código de barras obrigatório |
| Regras de Negócio | 1. Validar Api Key 2. Validar Bearer Token 3. Consultar protocolo na base de dados ABCOBPJ |
| Payload Request | api/v1/pagamentos/{codigo do cliente}/{codigo de barras} |
| Status Response | 200 |
| Header | Authorization = {Bearer}  ABC\_KeyID\_Pagamento = {key}  X-CorrelationID = {valor} |
| Payload Response | Pendente link Swagger |

# Referências

TBD...

# Pré-Requisito

TBD...

# Estimativa

TBD...