**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**

**PUC Minas Virtual**

**Pós-graduação *Lato Sensu* em Arquitetura de Soluções**

DOCUMENTO ARQUITETURAL DA PLATAFORMA DE PAGAMENTO GOOPAY

Rafael Augusto Miranda

Belo Horizonte

2023

**SUMÁRIO**

[1. INTRODUÇÃO 3](#_Toc141613954)

[2. CRONOGRAMA DO TRABALHO 6](#_Toc141613955)

[3. ESPECIFICAÇÃO ARQUITETURAL DA SOLUÇÃO 7](#_Toc141613956)

[3.1. Requisitos Funcionais 7](#_Toc141613957)

[3.2. Requisitos Não-funcionais 9](#_Toc141613958)

[3.3. Diagrama de Contexto 10](#_Toc141613959)

[4. MODELAGEM ARQUITETURAL 12](#_Toc141613960)

[4.1. Diagrama de Container 12](#_Toc141613961)

[4.2. Diagrama de Componentes 14](#_Toc141613962)

[4.2.1. Administração 14](#_Toc141613963)

[4.2.2. Gateway 16](#_Toc141613964)

[4.2.3. Backoffice 19](#_Toc141613965)

[4.2.4. Link de Pagamento 21](#_Toc141613966)

[4.2.5. Cartão de Crédito 23](#_Toc141613967)

[4.2.6. Cartão de Débito 25](#_Toc141613968)

[4.2.7. Boleto/BoletoPix 27](#_Toc141613969)

[4.2.8. Pix 29](#_Toc141613970)

[5. ARQUITETURA DE IMPLANTAÇÃO EM NUVEM 31](#_Toc141613971)

[6. AVALIAÇÃO DA ARQUITETURA (ATAM) 33](#_Toc141613972)

[6.1. Análise das abordagens arquiteturais 33](#_Toc141613973)

[6.2. Resultados Obtidos 33](#_Toc141613974)

[7. AVALIAÇÃO CRÍTICA DOS RESULTADOS 34](#_Toc141613975)

[8. CONCLUSÃO 35](#_Toc141613976)

[9. REFERÊNCIAS 36](#_Toc141613977)

# INTRODUÇÃO

Impulsionado pelo uso crescente de tecnologias digitais e móveis, o mercado de meios de pagamento passou por inúmeras transformações nos últimos tempos. As Fintechs surgiram e se destacaram nesse mercado, fornecendo soluções superiores aos bancos tradicionais. O comércio eletrônico aumentou a necessidade de soluções de pagamento rápidas e simplificadas, levando a avanços constantes neste setor. Além disso, na pandemia impulsionou ainda mais a adoção de métodos de pagamento digital, pois a segurança e a conveniência se tornaram as principais prioridades para muitos.

O mercado de meios de pagamento no Brasil é muito competitivo, com muitas empresas, nacionais e estrangeiras, lutando pelo domínio. Entre os muitos players estão Cielo, PagSeguro, Mercado Pago e Stone. As empresas de tecnologia têm ganhado o mercado oferecendo principalmente o Pix, opção de pagamento rápido desenvolvida pelo Banco Central do Brasil. Este mercado está em constante fluxo, portanto, acompanhar as últimas tendências e avanços emergentes na indústria é fundamental.

Já as empresas enfrentam uma constante luta para gerar lucro e manter-se competitivas em um mercado cada vez mais acirrado. Para obter sucesso financeiro, é necessário que elas tenham uma estratégia clara e bem definida, capaz de gerar valor para seus clientes e diferenciá-las de seus concorrentes. As empresas criam diversificação de produtos e serviços. Ao oferecer uma ampla gama de opções para seus clientes, as empresas podem aumentar suas receitas e reduzir sua dependência de um único produto ou serviço. Devido as empresas terem vários tipos de produtos e serviços, elas podem comercializar em vários tipos de canais como por exemplo loja, aplicativo, site, loja física.

A gestão financeira também é fundamental para a lucratividade das empresas. A adoção de práticas financeiras sólidas, pode ajudar a empresa a manter suas finanças saudáveis e garantir sua sobrevivência a longo prazo com desafios no dia a dia da gestão financeira usando diversos meios de pagamentos, entre os desafios são eles:

* Sem visibilidade da situação em tempo real (ou quase-real) das transações financeiras;
* Dificuldade para análise de ocorrências no processamento das transações;
* Grande esforço para criação de estudos para tomadas de decisão;
* Informações descentralizadas em diversas plataformas, dificultando a gestão;
* Dependência e concentração em poucos fornecedores.

Olhando na perspectiva tecnológica para disponibilizar um meio de pagamento em um canal de um negócio, era necessário um novo projeto de tecnologia entre a área de negócios e a área financeira. A figura 1 representa as necessidades das unidades de negócios em ter métodos de pagamento.

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente com confiança média

Figura 1 – Unidades de negócio por meios de pagamentos

A figura 02 representa os projetos de tecnologia para as unidades de negócio usando meios de pagamento, gerando redundância de projetos e tempo perdido da equipe tecnológica.

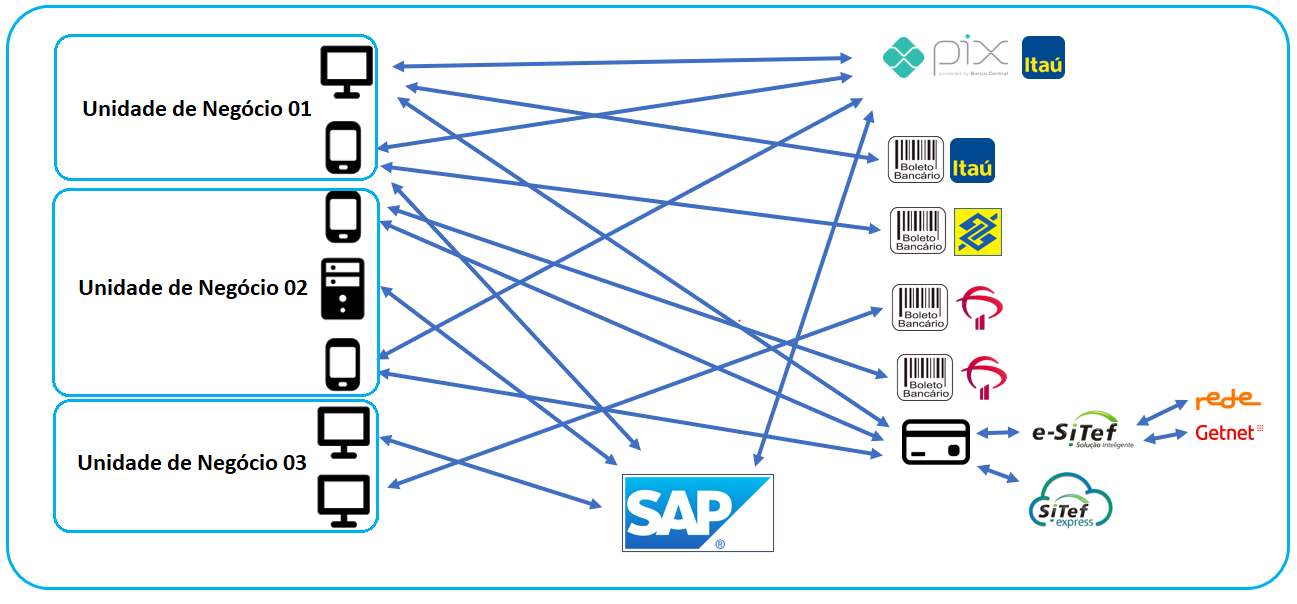


Figura 02 – Projetos de tecnologia redundantes

Devido a esse problema de visibilidade financeira, redundância de projetos de tecnologia em disponibilizar os meios de pagamento em canais de negócios, surge a proposta desse projeto que é o desenvolvimento de um único ponto onde os sistemas dos produtos de cada unidade de negócio da empresa possam se conectar e de acordo com regras de roteamento estabelecidas no sistema administrador, será roteado para um serviço que represente o tipo de pagamento para um provedor financeiro ou adquirente.

Este documento apresenta a descrição do projeto arquitetural da plataforma de orquestramento de meios de pagamento GooPay que atenda a todo um ecossistema de empresa de média e grande porte. A plataforma Goopay irá trazer uma série de benefícios:

* Um serviço de meio de pagamentos que atende todo o ecossistema da empresa;
* Fácil integração com qualquer nova aplicação;
* Os meios de pagamento são implementados somente uma vez, diretamente no serviço;
* Permite a troca das soluções de acordo com as negociações com instituições financeiras;
* Permite o uso simultâneo de soluções diferentes, estimulando a concorrência entre fornecedores;
* Traz contingência para as soluções de pagamentos;
* Possibilita maior controle da empresa na gestão de risco;
* Painel com informações para tomada de decisão;
* Ambiente seguro e aditável.

# CRONOGRAMA DO TRABALHO

A seguir é apresentado o cronograma proposto para as etapas deste trabalho.

|  |  |  |  |
| --- | --- | --- | --- |
| **Datas** | | **Atividade / Tarefa** | **Produto/Resultado** |
| **De** | **Até** |
| 11/03/2023 | 16/03/2023 | Introdução/Modelo Canvas/Cronograma do trabalho | Relatório Técnico |
| 17/03/2023 | 31/03/2023 | Requisitos Funcionais e Não-funcionais | Relatório Técnico |
| 02/04/2023 | 15/04/2023 | Diagrama de Contexto e Container | Relatório Técnico |
| 16/04/2023 | 30/04/2023 | Diagrama de Componentes | Relatório Técnico |
| 01/05/2023 | 14/05/2023 | Arquitetura de implantação em nuvem | Relatório Técnico |
| 15/05/2023 | 27/05/2023 | Avaliação da Arquitetura (ATAM) | Relatório Técnico |
| 28/05/2023 | 10/06/2023 | Análise das abordagens arquiteturais | Relatório Técnico |
| 11/06/2024 | 24/06/2023 | Resultados Obtidos | Relatório Técnico |
| 25/06/2023 | 08/07/2023 | Avaliação Crítica dos Resultados | Relatório Técnico |
| 09/07/2023 | 22/07/2023 | Conclusão | Relatório Técnico |

Tabela 1 - Cronograma do trabalho

# ESPECIFICAÇÃO ARQUITETURAL DA SOLUÇÃO

Esta seção apresenta a especificação básica da arquitetura da solução a ser desenvolvida, incluindo diagramas, restrições e requisitos definidos pelo autor, tal que permitem visualizar a macro arquitetura da solução.

## Requisitos Funcionais

Enumere todos os requisitos funcionais previstos para a sua aplicação. Lembre-se de listar todos os requisitos necessários para garantir cobertura arquitetural, segundo a especificação fornecida. Esta seção deve conter uma lista de requisitos visando produzir a modelagem completa da solução.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ID | Módulo | Descrição Resumida | Dificuldade | Prioridade |
| RF01 | Cartão de Crédito | O sistema deve permitir o pagamento por cartão de crédito. | Alta | Alta |
| RF02 | Cartão de Crédito | O sistema deve permitir o pagamento com 2 cartões ou mais cartões. | Alta | Média |
| RF03 | Cartão de Crédito | O sistema deve permitir pagamentos recorrentes. | Média | Baixa |
| RF04 | Cartão de Crédito | O sistema deve permitir retentativas de pagamentos para as falhas de transações. | Alta | Baixa |
| RF05 | Cartão de Crédito | O sistema deve permitir integração com empresa de antifraude. | Baixa | Alta |
| RF06 | Cartão de Crédito | O sistema deve permitir a verificação se o cartão é valido através do Zero dólar. | Baixa | Alta |
| RF07 | Cartão de Crédito | O sistema deve permitir a integração com conciliadoras. | Baixa | Baixa |
| RF08 | Gateway | O sistema deve permitir roteamento de transações de acordo com regras. | Baixa | Alta |
| RF09 | Gateway | O sistema deve permitir Multimeios de pagamento (cartões, cartão+pix, etc). | Alta | Baixa |
| RF10 | PIX | O sistema deve permitir o pagamento por PIX. | Média | Alta |
| RF11 | Boleto | O sistema deve permitir o pagamento por boleto. | Média | Alta |
| RF12 | Boleto | O sistema deve permitir o pagamento por boleto-pix | Média | Alta |
| RF13 | Cartão de débito | O sistema deve permitir o pagamento por cartão de débito. | Média | Alta |
| RF14 | Administração | O sistema deve permitir o cadastro empresas. | Baixa | Alta |
| RF15 | Administração | O sistema deve permitir o cadastro negócios. | Baixa | Alta |
| RF16 | Administração | O sistema deve permitir o cadastro canais. | Baixa | Alta |
| RF17 | Administração | O sistema deve permitir o cadastro meios de Pagamento. | Baixa | Alta |
| RF18 | Administração | O sistema deve permitir o cadastro de operadoras. | Baixa | Alta |
| RF19 | Administração | O sistema deve permitir a consulta de transações para auditorias. | Baixa | Baixa |
| RF20 | Administração | O sistema deve permitir o cadastro de regras de roteamento de transações. | Alta | Alta |
| RF21 | Backoffice | O sistema deve permitir a consulta de transações. | Média | Alta |
| RF22 | Backoffice | O sistema deve permitir o estorno de transações. | Alta | Baixa |
| RF23 | Backoffice | O sistema deve permitir a cancelamento de transações. | Média | Média |
| RF24 | Link de pagamento | O sistema deve permitir a geração de Link de pagamento customizado. | Alta | Alta |

Tabela 2 - Requisitos Funcionais

## Requisitos Não-funcionais

Enumere todos os requisitos não-funcionais previstos para a sua aplicação. Entre os requisitos não funcionais inclua todos os requisitos que julgar importantes do ponto de vista arquitetural, ou seja, os requisitos que terão impacto na definição da arquitetura. Os requisitos devem ser descritos de forma completa e preferencialmente.

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Descrição Resumida | Dificuldade | Prioridade |
| RNF1 | Dados sensíveis não devem ser armazenados no ambiente. | Baixa | Baixa |
| RNF2 | Todas as transações que passam pelo gateway de pagamento deveram ser armazenadas em banco de dados e em log. | Média | Alta |
| RNF3 | Teste de performance em ambiente deve aceitar 30 transações por segundo. | Alta | Alta |
| RNF4 | Acesso ao banco de dados do ambiente deverá ser feito através de usuários de sistema. | Baixa | Média |
| RNF5 | Ambiente disponível 24x7. | Alta | Média |
| RNF6 | O ambiente deve ser escalável, flexível e seguro. | Alta | Alta |
| RNF7 | O sistema deverá suportar três idiomas: portugês, espanhol e inglês. | Média | Média |
| RNF8 | O sistema deverá ser contruido com plataforma da Microsoft. | Média | Alta |
| RNF9 | O sistema deverá ter as cores da empresa. | Baixa | Alta |

Tabela 3 - Requisitos não funcionais.

## Diagrama de Contexto

[Diagrama

Descrição gerada automaticamente](https://raw.githubusercontent.com/raffaelmiranda/trabalho-conclusao-curso/main/PUC%20-%20Arquitetura%20de%20Solu%C3%A7%C3%B5es/Desenhos/Contexto.png)

Figura 3 - Diagrama de Contexto.

A [figura 3](https://raw.githubusercontent.com/raffaelmiranda/trabalho-conclusao-curso/main/PUC%20-%20Arquitetura%20de%20Solu%C3%A7%C3%B5es/Desenhos/Contexto.png) mostra o diagrama geral da solução proposta, com todos seus principais módulos e suas ligações.

A seguir vamos descrever os módulos internos do sistema:

* **Backoffice**: Fornece a visão consolidada das transações usando gráficos em dashboard para o gerente e analistas financeiros tomem decisões. O Backoffice tem o poder de estornar pagamentos dos clientes.
* **Administração**: Cadastros de empresas, negócios, produtos, canais, contratos e dados bancários, configurações de regras de roteamento dos métodos de pagamento diretamente na interface de administração.
* **Meios de Pagamento:** Recebe solicitações de pagamento através de link de pagamento e cobrança direta e faz roteamento de acordo com regras estabelecidas no sistema de administração para os métodos de pagamento estabelecidos nas adquirentes e instituições bancárias, após o processamento do pagamento utiliza o sistema de envio de e-mail para enviar o comprovante de pagamento para o cliente.
* **Link de Pagamento:** Recebe as informações para criar o link de pagamento, essas informações virão do sistema externo que faz a gestão do produto/serviço que o cliente utilizou.

A plataforma Goopay utiliza alguns componentes externos como o sistema de envio de e-mail para enviar o link de pagamento bem como o comprovante de pagamento para o cliente. Cada unidade de negócio tem o seu sistema que faz a sua gestão, esses sistemas se conectam com a plataforma Goopay para fazer uma cobrança direta como por exemplo no cartão de crédito do cliente bem como gerar o link de pagamento que será enviado para o cliente efetuar o pagamento da dívida em aberto.

Temos personas que podem usar o sistema Goopay que são o analista financeiro e o gerente financeiro. O analista e o gerente financeiro têm acesso ao Backoffice para fazer as análises das informações consolidadas através de dashboard, já o gerente financeiro tem acesso exclusivo no sistema administração para cadastrar empresas, negócios, produtos, canais, contratos e dados bancários, configurações de regras de roteamento para os métodos de pagamento.

# MODELAGEM ARQUITETURAL

Esta seção apresenta a modelagem arquitetural da solução proposta, de forma a permitir seu completo entendimento visando à implementação. O C4 Model é uma linguagem de modelagem arquitetural que ajuda a descrever sistemas de software de forma clara e concisa. Ele é composto por quatro níveis:

* Contexto: descreve o sistema e seu ambiente, incluindo os principais stakeholders e seus objetivos.
* Contêineres: descreve os principais componentes do sistema e suas interações.
* Componentes: descreve os principais módulos ou serviços que compõem cada contêiner.
* Código: descreve a implementação do sistema em nível de código.

## Diagrama de Container

A plataforma Goopay está divido em domínios de negócio em uma arquitetura por micro serviço com o banco de dados isolado e acessado somente pelo micro serviço do domínio que será utilizado. Os componentes estão distribuídos nos domínios de:

* BackOffice: Onde temos um frontend acessando o BFF de backoffice. O backoffice acessa o subdomínio gateway e o domínio administração para construir os dashboard de consulta e estorno de transações;
* Administração: Onde temos frontend acessando O BFF de administração. A administração é cadastrada os produtos, negócios, canais de vendas, empresas, informações de contratos com as instituições financeiras e adquirentes e as regras de roteamento para os meios de pagamentos;
* Link de pagamento: Onde temos o frontend acessando o BFF de link de pagamento. Contém os dados mínimos do cliente, dados da fatura, tipo de meios de pagamentos para ser escolhido a forma que o cliente que pagar, gerencia o ciclo de vida do link de pagamentos, envia as informações para o gateway e recebe as notificações sobre o processamento de pagamentos;
* Meios de pagamentos: Onde temos os subdomínios de:
  + Gateway: É responsável por processar os dados iniciais de pagamentos recebidos pelo link de pagamentos ou um pagamento direto feito pelos sistemas externos. O gateway acessa o domínio de administração para consultar as regras criadas para empresa e produtos/serviço consumidos. O gateway tem seu banco de dados exclusivo;
  + Cartão de Crédito: É responsável pela comunicação e processamento de pagamento com as adquirentes. A api de cartão de crédito tem seu banco de dados exclusivo;
  + Cartão de Débito: É responsável pela comunicação e processamento de pagamento com as instituições financeiras. A api de débito tem seu banco de dados exclusivo;
  + Boleto e Boleto Pix É responsável pela comunicação e processamento de pagamento com as instituições financeiras. A api de boleto/boleto pix tem seu banco de dados exclusivo;
  + Pix: É responsável pela comunicação e processamento de pagamento com as instituições financeiras. A api do pix tem seu banco de dados exclusivo.

A [figura 4](https://raw.githubusercontent.com/raffaelmiranda/trabalho-conclusao-curso/e5f8ab89412bda955553befab80f053acaeac0d6/PUC%20-%20Arquitetura%20de%20Solu%C3%A7%C3%B5es/Desenhos/Container.png) apresenta os containers da aplicação.

[Diagrama

Descrição gerada automaticamente](https://raw.githubusercontent.com/raffaelmiranda/trabalho-conclusao-curso/e5f8ab89412bda955553befab80f053acaeac0d6/PUC%20-%20Arquitetura%20de%20Solu%C3%A7%C3%B5es/Desenhos/Container.png)

Figura 4 - Diagrama de container.

## Diagrama de Componentes

Na seção iremos apresentar o diagrama de componentes da plataforma GooPay. O diagrama de componentes é um tipo de diagrama estrutural que é usado para representar a estrutura e as dependências entre os componentes de um sistema de software.

### Administração

[Diagrama, Esquemático

Descrição gerada automaticamente](https://raw.githubusercontent.com/raffaelmiranda/trabalho-conclusao-curso/main/PUC%20-%20Arquitetura%20de%20Solu%C3%A7%C3%B5es/Desenhos/ComponenteAdministracao.png)Figura 5 - Diagrama de Componentes - Administração

Conforme diagrama apresentado na [figura 5](https://raw.githubusercontent.com/raffaelmiranda/trabalho-conclusao-curso/main/PUC%20-%20Arquitetura%20de%20Solu%C3%A7%C3%B5es/Desenhos/ComponenteAdministracao.png), as entidades participantes da solução:

* Componente Externo
  + API Gateway: Api externa que consume o BFF Administração para consultar as regras criadas;
  + Web: Interface do sistema;
  + SGBD: Banco de dados SQL Server.
* Controller Company: Recebe os dados para cadastro das empresas;
* Controller Business: Recebe os dados para cadastro dos negócios que a empresa tem;
* Controller Product: Recebe os dados para cadastro dos produtos que um negócio tem;
* Controller Channel: Recebe os dados para cadastro do canal de venda que o produto/serviço é negociado;
* Controller Bank Data: Recebe os dados para cadastro dos dados bancários como instituição financeira com as contas bancárias e as contas que a cada negócio tem com adquirente de cartão de crédito;
* Controller Contract: Recebe os dados para cadastro dos contratos que a empresa negocia os produtos/serviços em algum canal de venda, bem como a vinculação de dados com as instituições financeiras e adquirentes;
* Controller Rule: Recebe os dados para cadastro de regras;
* Controller Rule Settings: Recebe os dados para cadastro das configurações das regras;
* Mediator: Padrão de projeto Mediator para redução de dependências entre as classes internas;
* Handler Business: Processa as informações recebidas para cadastro dos negócios que a empresa tem e salva no banco de dados;
* Handler Product: Processa as informações recebidas para cadastro dos produtos que um negócio tem e salva no banco de dados;
* Handler Channel: Processa as informações recebidas para cadastro do canal de venda que o produto/serviço é negociado e salva no banco de dados;
* Handler Company: Processa as informações recebidas para cadastro das empresas e salva no banco de dados;
* Handler Contract: Processa as informações recebidas para cadastro dos contratos que a empresa negocia os produtos/serviços em algum canal de venda, bem como a vinculação de dados com as instituições financeiras e adquirentes e salva no banco de dados;
* Handler Rule: Processa as informações recebidas para cadastro da regra. A formação da regra é contrato, produto, prioridade de execução da regra, se a regra está habilitada, se a regra foi publicada para o roteamento no gateway;
* Handler Rule Settings: Processa as informações recebidas para cadastro das configurações das regras. As configurações das regras são:
  + Faixa de montantes gerados: Valores de intervalos em reais;
  + Quantidade de transações geradas: Valores de intervalos em decimais;
  + Periodicidade de regra: Valores únicos em Diário, Semanal, Quinzenal, Mensal, Quadrimestral, Anual;
  + Faixa de montantes pagos: Valores de intervalos em reais;
  + Faixa de quantidades de pagamentos: Valores de intervalos em decimais;
  + Período de pagamento: Valores únicos em Diário, Semanal, Quinzenal, Mensal, Quadrimestral, Anual;
  + Período de atividade de regra: Valores de intervalos de data e hora.
* Handler Bank Data: Processa as informações recebidas para cadastro dos dados bancários como instituição financeira com as contas bancárias e as contas que a cada negócio tem com adquirente de cartão de crédito;
* Contexto de dados: Configurações de contexto do banco de dados, abstraindo qual é o banco de dados permitindo assim que se possa trocar o banco de dados por outro fabricante.

### Gateway

[Diagrama, Esquemático

Descrição gerada automaticamente](https://raw.githubusercontent.com/raffaelmiranda/trabalho-conclusao-curso/main/PUC%20-%20Arquitetura%20de%20Solu%C3%A7%C3%B5es/Desenhos/ComponenteGateway.png)

Figura 6 - Diagrama de Componentes - Gateway

Conforme diagrama apresentado na [figura 6](https://raw.githubusercontent.com/raffaelmiranda/trabalho-conclusao-curso/main/PUC%20-%20Arquitetura%20de%20Solu%C3%A7%C3%B5es/Desenhos/ComponenteGateway.png), as entidades participantes da solução:

* Componente Externo:
  + API BackOffice: Consome a API gateway para consultar as transações e cobranças;
  + API Link de Pagamento: Envia os dados básico de clientes, produtos/serviço consumido e escolha do tipo de pagamento pelo cliente para a API gateway;
  + Sistema Externo: Sistema da unidade de negócio que envia os dados básico de clientes, produtos/serviço consumido para o gateway efetuar uma cobrança direta no cliente;
  + API Cartão de Crédito: API que recebe os dados de cobrança e faz a comunicação com as adquirentes de acordo com as regras criadas no sistema administrador;
  + API Cartão de Débito: API que recebe os dados de cobrança e faz a comunicação com as instituições financeiras de acordo com as regras criadas no sistema administrador;
  + API Boleto/BoletoPix: API que recebe os dados de cobrança e faz a comunicação com as instituições financeiras de acordo com as regras criadas no sistema administrador;
  + API Pix: API que recebe os dados de cobrança e faz a comunicação com as instituições financeiras de acordo com as regras criadas no sistema administrador;
  + API Administrador: API gateway faz consultas das parametrizações de regras no sistema administrador;
  + Contexto de dados: Configurações de contexto do banco de dados, abstraindo qual é o banco de dados permitindo assim que se possa trocar o banco de dados por outro fabricante.
* Controller Views: Recebe os parâmetros para efetuar consultas de pagamentos, transações, cobrança para compor os gráficos no dashboard e estorno de pagamentos;
* Controller Charges: Recebe os dados através do sistema externo para criar a cobrança da dívida do cliente;
* Controller PaymentLink: Recebe os dados através do link de pagamento para criar a cobrança da dívida do cliente;
* Controller Synchronization: Recebe os dados através do sistema administrador com as informações que foram criadas;
* Mediator: Padrão de projeto Mediator para redução de dependências entre as classes internas;
* Handler Synchronization: Atualiza as informações das regras criadas no sistema administrador;
* Handler Transaction: Cria a transação do método de pagamento que será processado;
* Handler Customer: Cadastra as informações do cliente que está sendo cobrado;
* Handler Charge: Cadastra a cobrança do cliente;
* Handler Charge Itens: Cadastro os itens da cobrança que compõe o produto/serviço;
* Handler Payment Charge: Após identificar as regras o mesmo redireciona para a API especifica para do método de pagamento;
* Handler Parametrization: Busca as regras de roteamento de acordo com parâmetros de produto, empresa, método de pagamento e canal.

### Backoffice

[Diagrama, Esquemático

Descrição gerada automaticamente](https://raw.githubusercontent.com/raffaelmiranda/trabalho-conclusao-curso/main/PUC%20-%20Arquitetura%20de%20Solu%C3%A7%C3%B5es/Desenhos/ComponenteBackoffice.png)

Figura 7 - Diagrama de Componentes - Backoffice

Conforme diagrama apresentado na [figura 7](https://raw.githubusercontent.com/raffaelmiranda/trabalho-conclusao-curso/main/PUC%20-%20Arquitetura%20de%20Solu%C3%A7%C3%B5es/Desenhos/ComponenteBackoffice.png), as entidades participantes da solução:

* Componente Externo:
  + API Gateway: Consulta as cobranças e pagamento dos clientes;
  + Web: Interface do sistema.
* Controller Dashboard: Recebe o pedido para retornar as consolidações das informações para montagem de gráficos;
* Controller Conciliation: Recebe o pedido de download do relatório de conciliação enviados para empresas especializadas em conciliação bancária;
* Controller Charge: Recebe o pedido das listas de cobrança efetuadas dos clientes;
* Controller Payment: Recebe o pedido das listas de pagamento efetuados dos clientes;
* Mediator: Padrão de projeto Mediator para redução de dependências entre as classes internas;
* Handler Dashboard: Faz a consolidações das informações para montagem de gráficos;
* Handler Conciliation: Faz o download do relatório de conciliação enviados para empresas especializadas em conciliação bancária;
* Handler Charge: Retorna as listas de cobrança efetuadas dos clientes;
* Handler Payment: Retornas as listas de pagamento efetuados dos clientes.

### Link de Pagamento

[Diagrama, Esquemático

Descrição gerada automaticamente](https://raw.githubusercontent.com/raffaelmiranda/trabalho-conclusao-curso/main/PUC%20-%20Arquitetura%20de%20Solu%C3%A7%C3%B5es/Desenhos/ComponenteLinkDePagamento.png)

Figura 8 - Diagrama de Componentes - Link de Pagamento

Conforme diagrama apresentado na [figura 8](https://raw.githubusercontent.com/raffaelmiranda/trabalho-conclusao-curso/main/PUC%20-%20Arquitetura%20de%20Solu%C3%A7%C3%B5es/Desenhos/ComponenteLinkDePagamento.png), as entidades participantes da solução:

* Componente Externo:
  + API Gateway: BFF do link de pagamento envia os dados para a API Gateway;
  + Sistema Externo: Solicita para o BFF do link de pagamento para gerar o link de pagamentos com os dados básicos da cobrança;
  + Sistema de envio de e-mail: Recebe a solicitação para enviar o link de pagamento para e-mail do cliente;
  + Web: Interface do link de pagamento;
  + SGBD: Banco de dados SQL Server.
* Controller Payment: Recebe os dados para pagamento direto;
* Controller Generate Link: Recebe os dados para pagamento pelo link de pagamento;
* Mediator: Padrão de projeto Mediator para redução de dependências entre as classes internas;
* Handler Gateway: Processa as informações recebidas e repassa para o gateway decidir a forma de pagamento de acordo com regras estabelecidas
* Handler Generate Link: Gera o link de pagamento para ser enviado para os clientes. Identifica que as informações de um novo pagamento pelo link de pagamento e envia as informações para o componente gateway;
* Contexto de dados: Configurações de contexto do banco de dados, abstraindo qual é o banco de dados permitindo assim que se possa trocar o banco de dados por outro fabricante.

### Cartão de Crédito

[Diagrama, Esquemático

Descrição gerada automaticamente](https://raw.githubusercontent.com/raffaelmiranda/trabalho-conclusao-curso/main/PUC%20-%20Arquitetura%20de%20Solu%C3%A7%C3%B5es/Desenhos/ComponenteCredito.png)

Figura 9 - Diagrama de Componentes - Cartão de Crédito

Conforme diagrama apresentado na [figura 9](https://raw.githubusercontent.com/raffaelmiranda/trabalho-conclusao-curso/main/PUC%20-%20Arquitetura%20de%20Solu%C3%A7%C3%B5es/Desenhos/ComponenteCredito.png), as entidades participantes da solução:

* Componente Externo:
  + API Adquirentes: API que fornece a interface para o pagamento ser por cartão de crédito;
  + SGBD: Banco de dados SQL Server
  + API Gateway: Recebe os dados da API Gateway para efetuar o pagamento da cobrança.
* Controller Payment: Recebe as informações e repassa para o handler específico para identificar qual adquirente que deve ser chamado para processar a informação salvando os dados no banco de dados;
* Controller RefundPayment: Recebe as informações e repassa para o handler específico para identificar qual adquirente que deve ser chamado para efetuar o estorno da transação;
* Mediator: Padrão de projeto Mediator para redução de dependências entre as classes internas;
* Handler MakePayment: Identifica qual adquirente que deve ser chamado para processar a informação salvando os dados no banco de dados;
* Handler RefundPayment: Identifica qual adquirente que deve ser chamado para efetuar o estorno da transação;
* Handler Rede: Faz a chamada para API da Rede para efetuar a cobrança;
* Hanlder Cielo: Faz a chamada para API da Cielo para efetuar a cobrança;
* Handler Getnet: Faz a chamada para API da Getnet para efetuar a cobrança;
* Contexto de dados: Configurações de contexto do banco de dados, abstraindo qual é o banco de dados permitindo assim que se possa trocar o banco de dados por outro fabricante.

### Cartão de Débito

[Diagrama, Esquemático

Descrição gerada automaticamente](https://raw.githubusercontent.com/raffaelmiranda/trabalho-conclusao-curso/main/PUC%20-%20Arquitetura%20de%20Solu%C3%A7%C3%B5es/Desenhos/ComponenteDebito.png)

Figura 10 - Diagrama de Componentes - Cartão de Débito

Conforme diagrama apresentado na [figura 10](https://raw.githubusercontent.com/raffaelmiranda/trabalho-conclusao-curso/main/PUC%20-%20Arquitetura%20de%20Solu%C3%A7%C3%B5es/Desenhos/ComponenteDebito.png), as entidades participantes da solução:

* Componente Externo:
  + API Instituições financeiras: API que fornece a interface para o pagamento ser por cartão de débito;
  + SGBD: Banco de dados SQL Server
  + API Gateway: Recebe os dados da API Gateway para efetuar o pagamento da cobrança.
* Controller Payment: Recebe as informações e repassa para o handler específico para identificar qual adquirente que deve ser chamado para processar a informação salvando os dados no banco de dados;
* Controller RefundPayment: Recebe as informações e repassa para o handler específico para identificar qual instituição financeira que deve ser chamado para efetuar o estorno da transação;
* Mediator: Padrão de projeto Mediator para redução de dependências entre as classes internas;
* Handler MakePayment: Identifica qual instituição financeira que deve ser chamado para processar a informação salvando os dados no banco de dados;
* Handler RefundPayment: Identifica qual instituição financeira que deve ser chamado para efetuar o estorno da transação;
* Handler Santander: Faz a chamada para API do Santander para efetuar a cobrança;
* Hanlder Itaú: Faz a chamada para API do Itaú para efetuar a cobrança;
* Handler Banco do Brasil: Faz a chamada para API do Banco do Brasil para efetuar a cobrança;
* Contexto de dados: Configurações de contexto do banco de dados, abstraindo qual é o banco de dados permitindo assim que se possa trocar o banco de dados por outro fabricante.

### Boleto/BoletoPix

[Diagrama, Esquemático

Descrição gerada automaticamente](https://raw.githubusercontent.com/raffaelmiranda/trabalho-conclusao-curso/main/PUC%20-%20Arquitetura%20de%20Solu%C3%A7%C3%B5es/Desenhos/ComponenteBoletoBoletoPix.png)

Figura 11 - Diagrama de Componentes - Boleto/BoletoPix

Conforme diagrama apresentado na [figura 11](https://raw.githubusercontent.com/raffaelmiranda/trabalho-conclusao-curso/main/PUC%20-%20Arquitetura%20de%20Solu%C3%A7%C3%B5es/Desenhos/ComponenteBoletoBoletoPix.png), as entidades participantes da solução:

* Componente Externo:
  + API Instituições financeiras: API que fornece a interface para o pagamento ser por boleto/boleto pix;
  + SGBD: Banco de dados SQL Server
  + API Gateway: Recebe os dados da API Gateway para efetuar o pagamento da cobrança.
* Controller Payment: Recebe as informações e repassa para o handler específico para identificar qual instituição financeira que deve ser chamado para processar a informação salvando os dados no banco de dados;
* Controller RefundPayment: Recebe as informações e repassa para o handler específico para identificar qual instituição financeira que deve ser chamado para efetuar o estorno da transação;
* Mediator: Padrão de projeto Mediator para redução de dependências entre as classes internas;
* Handler MakePayment: Identifica qual instituição financeira que deve ser chamado para processar a informação salvando os dados no banco de dados;
* Handler RefundPayment: Identifica qual instituição financeira que deve ser chamado para efetuar o estorno da transação;
* Handler Santander: Faz a chamada para API do Santander para efetuar a cobrança;
* Hanlder Itaú: Faz a chamada para API do Itaú para efetuar a cobrança;
* Handler Banco do Brasil: Faz a chamada para API do Banco do Brasil para efetuar a cobrança;
* Contexto de dados: Configurações de contexto do banco de dados, abstraindo qual é o banco de dados permitindo assim que se possa trocar o banco de dados por outro fabricante.

### Pix

[Diagrama, Esquemático

Descrição gerada automaticamente](https://raw.githubusercontent.com/raffaelmiranda/trabalho-conclusao-curso/main/PUC%20-%20Arquitetura%20de%20Solu%C3%A7%C3%B5es/Desenhos/ComponentePix.png)

Figura 12 - Diagrama de Componentes - Boleto/BoletoPix

Conforme diagrama apresentado na [figura 12](https://raw.githubusercontent.com/raffaelmiranda/trabalho-conclusao-curso/main/PUC%20-%20Arquitetura%20de%20Solu%C3%A7%C3%B5es/Desenhos/ComponentePix.png), as entidades participantes da solução:

* Componente Externo:
  + API Instituições financeiras: API que fornece a interface para o pagamento ser por pix;
  + SGBD: Banco de dados SQL Server
  + API Gateway: Recebe os dados da API Gateway para efetuar o pagamento da cobrança.
* Controller Payment: Recebe as informações e repassa para o handler específico para identificar qual instituição financeira que deve ser chamado para processar a informação salvando os dados no banco de dados;
* Controller RefundPayment: Recebe as informações e repassa para o handler específico para identificar qual instituição financeira que deve ser chamado para efetuar o estorno da transação;
* Mediator: Padrão de projeto Mediator para redução de dependências entre as classes internas;
* Handler MakePayment: Identifica qual instituição financeira que deve ser chamado para processar a informação salvando os dados no banco de dados;
* Handler RefundPayment: Identifica qual instituição financeira que deve ser chamado para efetuar o estorno da transação;
* Handler Santander: Faz a chamada para API do Santander para efetuar a cobrança;
* Hanlder Itaú: Faz a chamada para API do Itaú para efetuar a cobrança;
* Handler Banco do Brasil: Faz a chamada para API do Banco do Brasil para efetuar a cobrança;
* Contexto de dados: Configurações de contexto do banco de dados, abstraindo qual é o banco de dados permitindo assim que se possa trocar o banco de dados por outro fabricante.

# ARQUITETURA DE IMPLANTAÇÃO EM NUVEM

## Infraestrutura na nuvem

A plataforma Goopay será hospedado na Cloud Azure da Microsoft, o desenho logo abaixo representa a arquitetura de infraestrutura.

[Gráfico, Diagrama, Esquemático, Gráfico de caixa estreita

Descrição gerada automaticamente](https://raw.githubusercontent.com/raffaelmiranda/trabalho-conclusao-curso/main/PUC%20-%20Arquitetura%20de%20Solu%C3%A7%C3%B5es/Desenhos/ArquiteturaAzure.png)

Figura 13 - Diagrama dos recursos alocados

Os nomes dos recursos seguem as melhores práticas definidos pela Microsoft usando o [Framework Cloud Adoption](https://learn.microsoft.com/pt-br/azure/cloud-adoption-framework/ready/azure-best-practices/resource-naming).

Recursos utilizado na construção da plataforma:

* Grupo de recursos
* Plano do Serviço de Aplicativo
* Serviço de Aplicativo
* SQL Server
* Banco de dados SQL
* Application Insights
* Ponto de extremidade privado
* Adaptador de Rede
* Rede Virtual

Os serviços de aplicativo foram distribuídos em vários planos do serviço de aplicativo para distribuição de carga e separação de contextos. A rede virtual permite que todos os recursos se comuniquem internamente dentro de uma rede privada. Applications Insights é um serviço de monitoramento de desempenho de aplicativos. Grupo de recursos é um contêiner que mantém os recursos relacionados a uma solução do Azure. SQL Server é um sistema gerenciador de banco de dados. Um banco de dados no SQL Server é composto de uma coleção de tabelas que armazenam um conjunto específico de dados estruturados.

[Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente](https://raw.githubusercontent.com/raffaelmiranda/trabalho-conclusao-curso/main/PUC%20-%20Arquitetura%20de%20Solu%C3%A7%C3%B5es/Desenhos/RecursosAzure.png)

Figura 14 - Recursos Criados

## Fluxo de dados do pipeline

A plataforma Goopay será irá utilizar a plataforma Azure Devops da Microsoft para gerenciar todo o ciclo de desenvolvimento, a [figura](https://raw.githubusercontent.com/raffaelmiranda/trabalho-conclusao-curso/main/PUC%20-%20Arquitetura%20de%20Solu%C3%A7%C3%B5es/Desenhos/Pipeline.png) logo abaixo representa o fluxo de dados do pipeline.

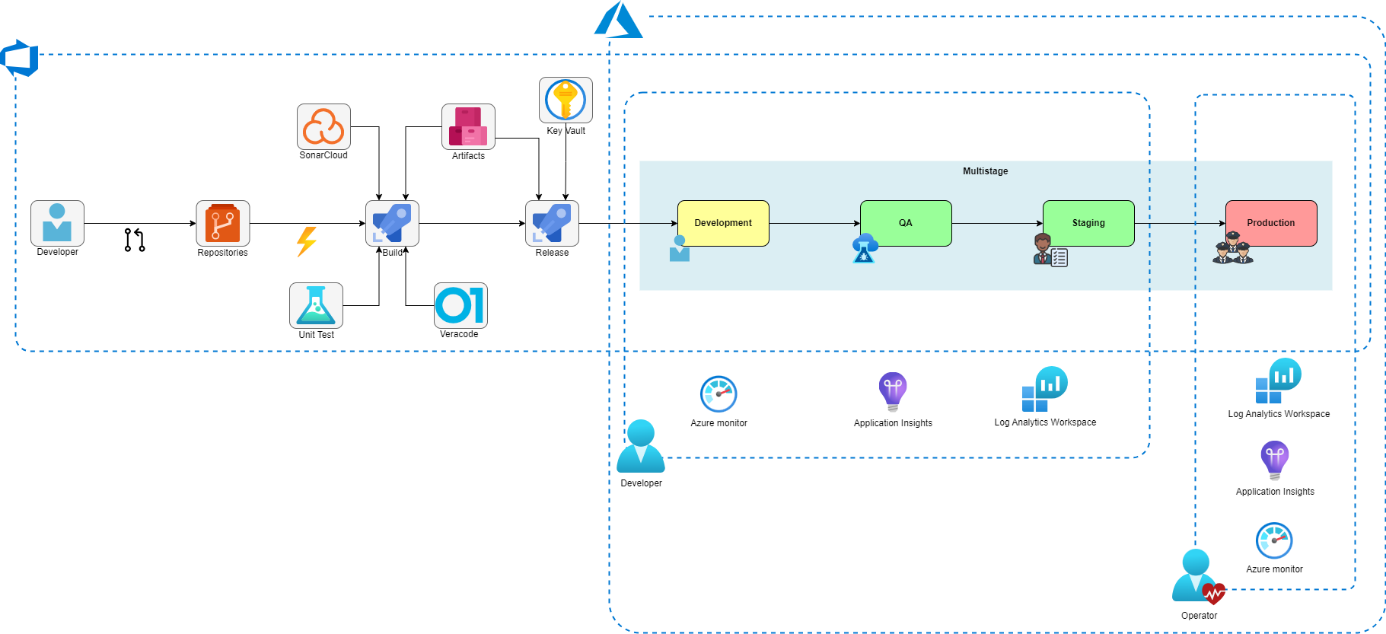
[](https://raw.githubusercontent.com/raffaelmiranda/trabalho-conclusao-curso/main/PUC%20-%20Arquitetura%20de%20Solu%C3%A7%C3%B5es/Desenhos/Pipeline.png)

Figura 15 - Diagrama de fluxo de dados

O fluxo de dados acontece na seguinte maneira:

Após o desenvolvedor terminar a codificação, ele solicita uma PR para ser revisado por outro desenvolvedor. O segundo desenvolvedor indica as melhorias que devem ser feitas caso encontre, senão a PR é integrada na branch de develop e irá disparar automático (continuous integration) o build do novo código fonte integrado.

[Tela de celular com publicação numa rede social

Descrição gerada automaticamente](https://raw.githubusercontent.com/raffaelmiranda/trabalho-conclusao-curso/main/PUC%20-%20Arquitetura%20de%20Solu%C3%A7%C3%B5es/Desenhos/build-trigger.png)

Figura 16 - Continuous Integration habilitado

Durante o build da branch develop ocorre a vários processos de verificação de qualidade de código, execução de testes de unidade, scanner de código fonte para detecção de vulnerabilidade. Todas as ferramentas estarão configuradas com métricas e somente irá promover para o ambiente developer com cobertura de teste de no mínimo com 80% de cobertura. Caso não atinja 80% o build.

Funções utilitárias serão compartilhadas entre os projetos de API’s esse um projeto do tipo library e será hospedada no repositório de pacotes Azure Artifacts. Durante o processo de build e release o pipeline irá fazer download dos artefatos para compor o pacote que será implantado posteriormente.

Durante o release o pipeline irá baixar os artefatos do build criados no pipeline de CI e irá exigir as credenciais guardadas no Azure Key Vault que exigem processo de autenticação nas API’s externas da plataforma Goopay. O pipeline de CD irá implantar no ambiente de developement.

O desenvolvedor irá efetuar teste mínimos no ambiente de desenvolvimento antes de liberar a release para o ambiente de QA. O time de QA irá efetuar todos os devidos testes para garantir a qualidade antes de liberar a versão da release no ambiente de Staging. Caso o time de QA encontre inconformidades o time de QA abre um card do tipo Bug no Azure Boards com todas as evidências para a atuação do desenvolvedor. O desenvolvedor efetua a correção e o processo se repete até prover uma nova versão de release. Caso time de QA não encontre inconformidades o próprio time libera a versão da release em ambiente de Staging.

O Product Owner realiza a homologação das novas features no ambiente de Staging. A feature estando corretamente desenvolvida, o product owner libera a versão de release no ambiente de Production.

No monitoramento de ambientes, o ambiente de developement, QA e staging compartilham os mesmos recursos de monitoramento, ambiente de production utiliza de forma isolada o monitoramento do ambiente. A tríade de recursos que são utilizados no monitoramento são o Azure Monitor que coleta dados como logs métricas, telemetria de aplicativos e métricas de plataforma para os serviços do Azure, o Application Insights coleta dados de monitoramento específicos do aplicativo em tempo real sobre o desempenho e uso e o Azure Log Analytics é usado para armazenar central onde pode armazenas consultar e analisar dados de várias fontes, incluindo recursos, aplicativos e serviços do Azure.

# AVALIAÇÃO DA ARQUITETURA (ATAM)

Deve-se apresentar pelo menos **duas evidências** de cada avaliação realizada, sendo uma textual e outra como figura.

## Análise das abordagens arquiteturais

apresente evidências (textuais e/ou de imagens) que comprovem o atendimento da aplicação aos cenários submetidos a avaliação, segundo o método ATAM;

Apresente aqui um breve resumo das principais características da proposta arquitetural. Para isto, utilize o método Architecture Tradeoff Analysis Method (ATAM), no qual são utilizados cenários para fazer essa análise.

Exemplo:

|  |  |  |  |
| --- | --- | --- | --- |
| **Atributos de Quailidade** | **Cenários** | **Importância** | **Complexidade** |
| Interoperabilidade | Cenário 1: O sistema deve se comunicar com sistemas de outras tecnologias. | A | M |
| Usabilidade | Cenário 2: O sistema deve prover boa usabilidade. | M | B |
| Manutenibilidade | Cenário 3: O sistema deve ter a manutenção facilitada. | M | M |

## Resultados Obtidos

apresente, num quadro, um resumo com os resultados que foram alcançados pelo projeto, em termos dos requisitos não-funcionais almejados;

Apresente os resultados da arquitetura produzida, indicando seus pontos fortes e suas limitações. A título de sugestão construa uma tabela apresentando esses resultados, como no exemplo que segue:

|  |  |  |
| --- | --- | --- |
| **Requisitos Não Funcionais** | **Teste** | **Homologação** |
| RNF01: O sistema deve ... | OK | OK |
| RNF02: O sistema deve ... | OK | N.A. |
| RNF03: ... | OK | N.A. |

# AVALIAÇÃO CRÍTICA DOS RESULTADOS

escreva uma avaliação crítica, refletindo sobre o aprendizado, as dificuldades encontradas, prós e contras. É importante cotejar com os resultados almejados, visando verificar o que se conseguiu e o que não foi possível atingir, em termos de objetivos;

O texto produzido deve mostrar visão crítica do aluno, descrevendo prós e contras da sua arquitetura.

Apresente aqui, de forma resumida, os principais pontos positivos e negativos da arquitetura proposta. Adote uma postura crítica que permita entender as limitações arquiturais, incluindo os prós e contras das tecnologias. Você pode utilizar o formato textual ou produzir um quadro resumo.

Ex. de quadro resumo:

|  |  |
| --- | --- |
| **Ponto avaliado** | **Descrição** |
| xxxxxxxxxxxxxxx | xxxxxxxxxxxxxxxxxxxx |
|  |  |
|  |  |

# CONCLUSÃO

escreva uma conclusão de modo a encerrar o trabalho, mostrando também as possibilidades para sua continuação futura**.**

As conclusões apresentadas devem ser sustentáveis, sendo bem embasadas, e estar de acordo com o trabalho realizado.

Descreva, de forma sucinta, quais foram as lições aprendidas na execução do seu projeto arquitetural. Procure apresentá-las de tal forma que fiquem configurados os *trade-offs* da arquitetura produzida, como por exemplo, Segurança X Desempenho, Granularidade X Manutenibilidade, etc.

Aqui deve ser apresentado também tudo que se aprendeu com esse projeto, de modo a servir como ajuda para outros profissionais.

Também se faz necessário evidenciar as possibilidades de melhoria do projeto, caso se deseje dar continuidade a ele. Nesse sentido, indique possíveis ajustes ou melhorias arquiteturais, que possam vir a ser realizados futuramente.

Lições aprendidas (ex.):

1. xxxxxxxxxxxxxxxxx
2. xxxxxxxxxxxxxxxxx
3. xxxxxxxxxxxxxxxxx

# REFERÊNCIAS

T. A Guedes, Gilleanes. UML 2 - Uma abordagem Prática 3ª Edição**.** Cidade: São Paulo Novatec Editora Ltda, 2018.

Meios de pagamento: o que são, quais os tipos e as oportunidades de mercado na América Latina. Fluid Blog, 2023. Disponível em: <https://dock.tech/blog/meios-de-pagamento/>. Acesso em: 10/04/2023

Mercado de pagamentos: o que é e como funciona? Instituto Propague, 2023. Disponível em: [https://institutopropague.org/pagamentos/mercado-de-pagamentos-o-que-e-e-como-funciona/.](https://institutopropague.org/pagamentos/mercado-de-pagamentos-o-que-e-e-como-funciona/.%20) Acesso em: 10/04/2023

A evolução do mercado de meios de pagamento brasileiro. RTM, 2023. Disponível em: <https://www.rtm.net.br/a-evolucao-do-mercado-de-meios-de-pagamento-brasileiro/>. Acesso em: 10/04/2023

Tudo o que você precisa saber sobre o mercado de pagamentos. Blog da Galícia Educação, 2023. Disponível em: <https://www.galiciaeducacao.com.br/blog/tudo-o-que-voce-precisa-saber-sobre-o-mercado-de-pagamentos/>. Acesso em: 10/05/2023

Confira quatro tendências no mercado de pagamentos da América Latina. Mercado Consumo, 2023. Disponível em: <https://mercadoeconsumo.com.br/24/01/2023/meios-de-pagamento/confira-quatro-tendencias-no-mercado-de-pagamentos-da-america-latina/>. Acesso em: 10/05/2023

PIOVESANA, MATHEUS. Pix foi o meio de pagamento mais utilizado no Brasil em 2022. Einvestidor, 2023. Disponível em: <https://einvestidor.estadao.com.br/ultimas/meios-pagamento-mais-usados-2022/>. Acesso em: 10/04/2023

Gerenciar grupos de recursos do Azure usando o portal do Azure, Microsoft Learn, 2023. Disponível em: <https://learn.microsoft.com/pt-br/azure/azure-resource-manager/management/manage-resource-groups-portal>. Acesso em: 16/07/2023

Visão geral do plano do Serviço de Aplicativo do Azure, Microsoft Learn, 2023. Disponível em: <https://learn.microsoft.com/pt-br/azure/app-service/overview-hosting-plans>. Acesso em: 16/07/2023

Gerenciar um plano do Serviço de Aplicativo no Azure, Microsoft Learn, 2023. Disponível em: <https://learn.microsoft.com/pt-br/azure/app-service/app-service-plan-manage>. Acesso em: 16/07/2023

Obtenha observabilidade de ponta a ponta para seus aplicativos, infraestrutura e rede, Azure Monitor, 2023. Disponível em: <https://azure.microsoft.com/services/monitor>. Acesso em: 30/07/2023

Visão geral do Application Insights, Microsoft Learn, 2023. Disponível em: <https://learn.microsoft.com/pt-br/azure/azure-monitor/app/app-insights-overview?tabs=net>. Acesso em: 30/07/2023

Visão Geral do workspace do Log Analytics, Microsoft Learn, 2023. Disponível em: <https://learn.microsoft.com/pt-br/azure/azure-monitor/logs/log-analytics-workspace-overview>. Acesso em: 30/07/2023