INSTITUTO INFNET

PÓS-GRADUAÇÃO MIT ENGENHARIA DE SOFTWARE .NET

ALEXANDER VIEIRA DA SILVA

MARLON DANTAS BRAGA

PEDRO FERNANDO ROCHA NOVAES

PROJETO DE BLOCO

PROFESSOR TOMÁS DE AQUINO

RIO DE JANEIRO-RJ

PROJETO DE BLOCO

Utilização de requisitos funcionais, não funcionais e regras de negócio

Requisitos de alto nível de sistema (RANS)

Os RANs são descrições do que o sistema/produto deve ou não fazer. Eles não definem como deve ser feito, pois isto já compete a área técnica. Eles geralmente se concentram nos resultados desejados e nas funcionalidades gerais que o sistema ou produto deve ter para atender às necessidades dos usuários e das partes interessadas.

- O sistema deve permitir que os usuários façam login e gerenciem suas contas.
- O produto deve ser fácil de usar para usuários com habilidades limitadas em tecnologia.
- O sistema deve ser capaz de processar um grande volume de transações em tempo real.
- O produto deve ser seguro e proteger os dados confidenciais dos usuários.
- O sistema deve ser escalável e capaz de lidar com o crescimento futuro da empresa.
- O Produto enviado deverá ser rastreável.

O aluno escreveu e apresentou os Requisitos Funcionais (RF)

Requisitos Funcionais	
[RF-001]	Manter Produto
[RF-002]	Manter Estoque
[RF-003]	Manter Categoria
[RF-004]	Gerir venda
[RF-005]	Aplicar voucher pedido
[RF-006]	Estornar estoque

Requisito	Regra de negócio
RF 01	QUANTIDADE: produto é considerado disponível quando está ativo e tem estoque > 0
RF 01	PRODUTO: tem que ter obrigatoriamente id, nome, descrição, valor e imagem
RF 01	PRODUTO: dimensões devem ter altura, largura e profundidade maior ou igual a zero
RF 01	PRODUTO: a formatação das dimensões deve ser "L x A x P"
RF 01	PRODUTO: Sistema deve exibir detalhes de produto
RF 02	ESTOQUE: Ao concluir uma compra, sistema deve debitar quantidade de itens de estoque
RF 03	PRODUTO: Sistema deve listar todos os produtos disponíveis
RF 03	PRODUTO: Sistema deve listar produtos disponíveis filtrados por categoria
RF 03	CATEGORIA: deve ter obrigatoriamente Nome e código
RF 04	PAGAMENTO: Pagamento pode ser realizado por cartão de crédito e boleto
RF 04	CARRINHO: Precisa ter 1 ou mais itens para ser fechado
RF 04	PAGAMENTO: Status de transação pode ser Pago ou Recusado
RF 04	COMPRA: Sistema deve recuperar resumo de compra desejado
RF 05	VOUCHER: Carrinho de compras - Desconto absoluto e/ou percentual.
RF 06	PAGAMENTO: Se o pagamento não for realizado com sucesso, a compra não deve ser concluída
RF 06	ESTORNO: Em casos de devolução de produto, estoque deve ser acrescido da quantidade devolvida

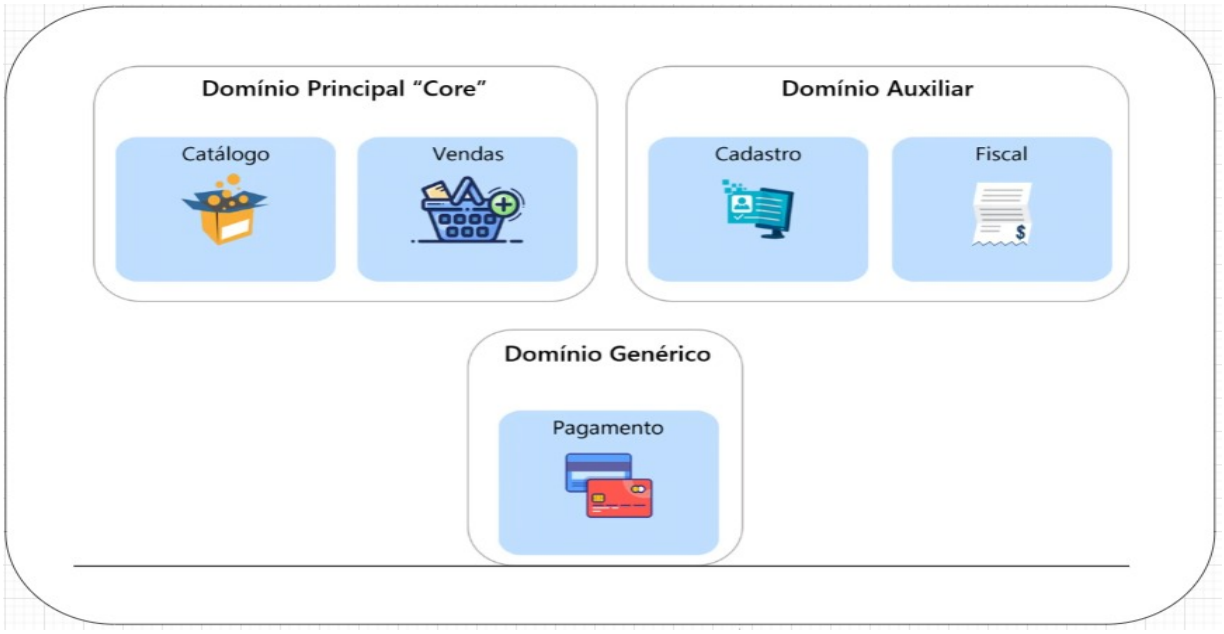
O aluno escreveu e apresentou os Requisitos Não Funcionais (RNF)

Código	Requisitos Não Funcionais	Indicadores
RNF 01	Desempenho	Tempo de Resposta das requisições(1 a cada 100 ms)
RNF 02	Disponibilidade	Funcional em tempo integral (24h/7 dias por semana)
RNF 03	Escalabilidade	picos de consumo de até 1k acessos/min
RNF 04	Segurança	Autenticação JWT / criptografia Char 256
RNF 05	Privacidade	Cumprimento da LGPD

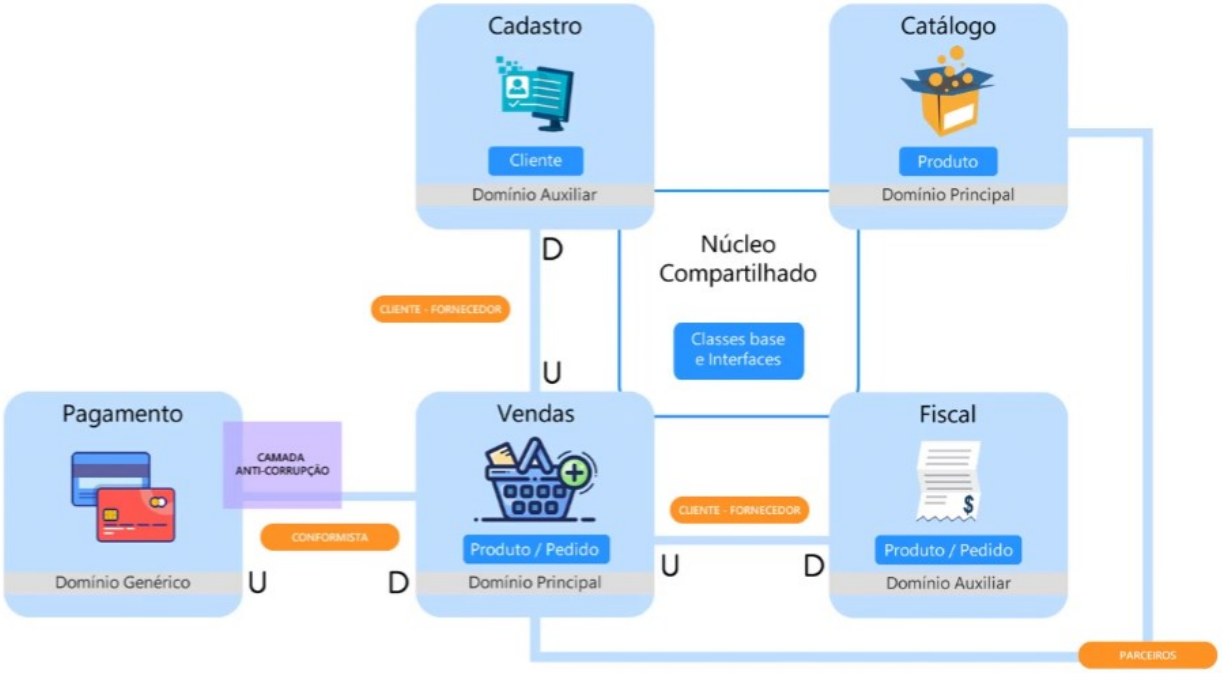
Aplicar os conceitos de DDD

O aluno apresentou o modelo de domínio

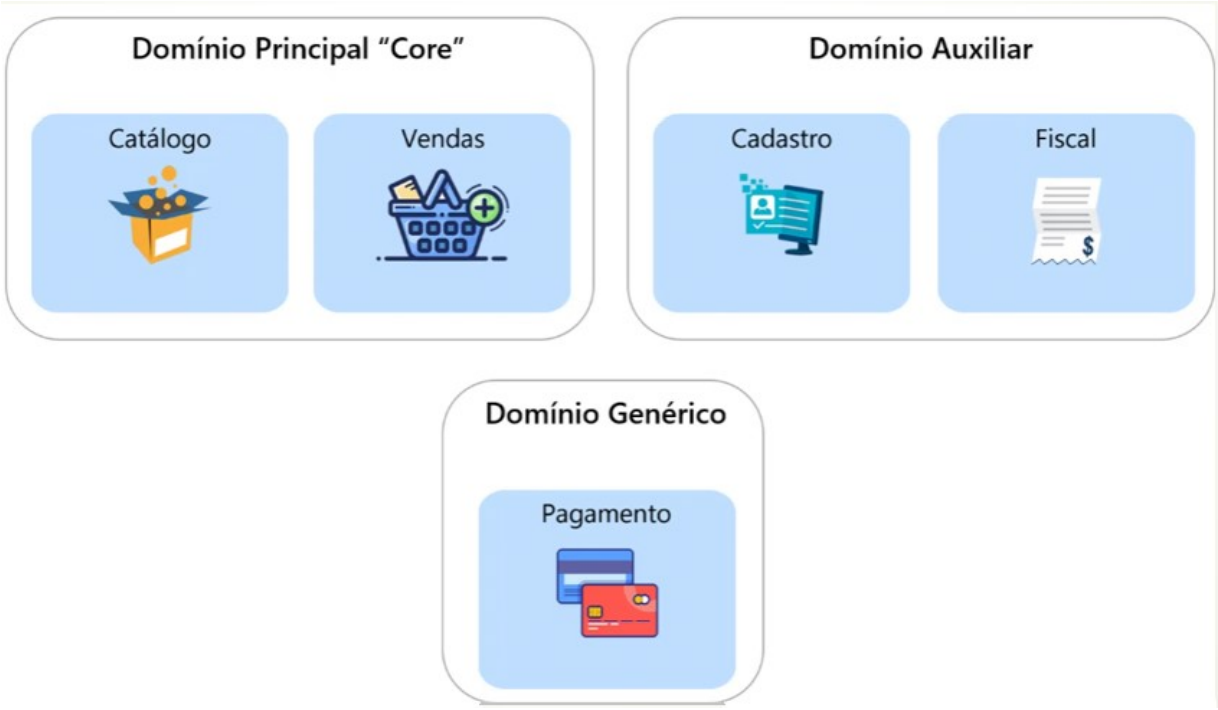
Domínio - NerdStore



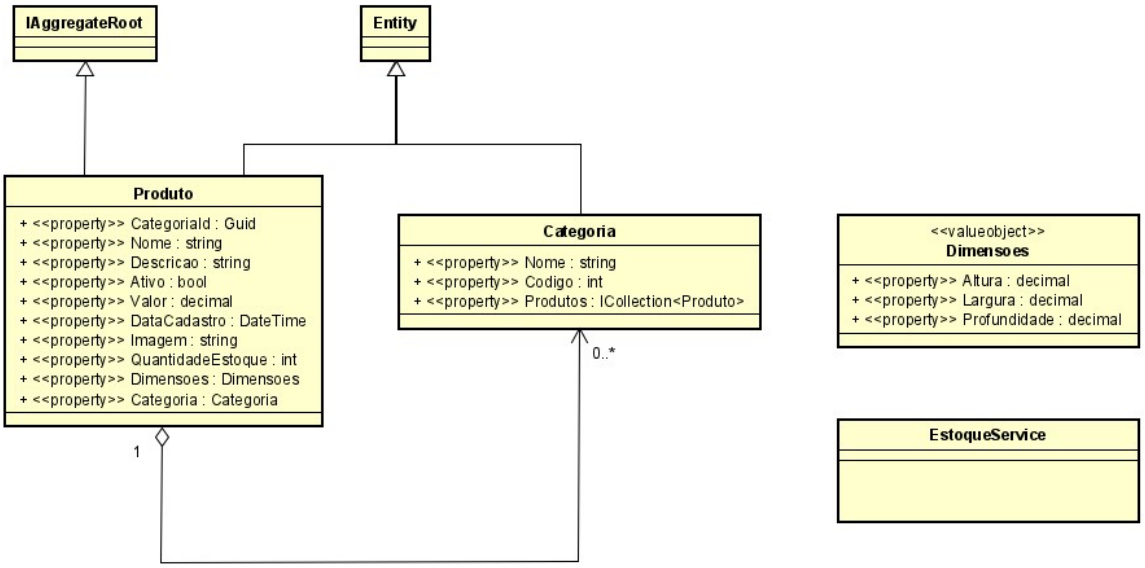
Context Map



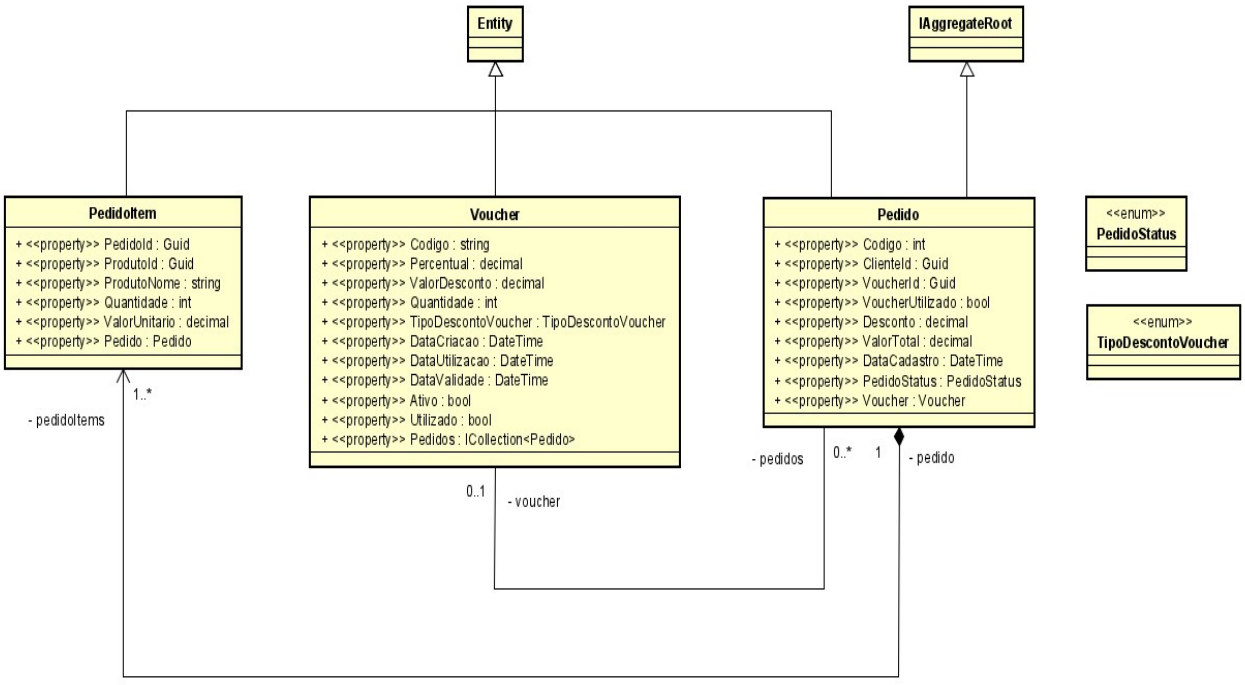
O aluno apresentou subdomínios



O aluno apresentou value objects



O aluno apresentou pelo menos um agregado



Utilizar Modelagem de Software

O aluno apresentou Matriz de Rastreabilidade (requisitos funcionais X casos de uso)

Requesitos X Casos de uso		[UC-001] Obter Produto por Categoria	[UC-002] Obter Produto por Id	[UC-003] Obter Todos os Produtos	[UC-004] Adicionar Produto	[UC-005] Atualizar Produto	[UC-006] Debitar Produto em Estoque	[UC-007] Repor Produto em Estoque	[UC-008] Debitar Estoque	[UC-009] Debitar Lista Produtos Pedido	[UC-010] Repor Estoque	[UC-011] Repor Lista Produtos Pedido	[UC-012] Obter Categorias	[UC-013] Adicionar item pedido	[UC-014] Cancelar processamento pedido	[UC-015] Finalizar pedido	[UC-015] Inicar pedido	[UC-015] Remove item pedido	[UC-015] Obtem pagamento
[RF-001]	Manter Produto	X	X	X	X	X													
[RF-002]	Manter Estoque						X	X	X	X	X	X							
[RF-003]	Manter Categoria												X						
[RF-004]	Gerir venda													X					
[RF-005]	Aplicar voucher pedido														X				
[RF-006]	Estornar estoque															X	X	X	X

O aluno apresentou métrica de estimativa de prazo para conclusão do sistema

Antes de mensurar a estimativa de custo do projeto. É necessário estimar quanto recurso exigirá seu desenvolvimento. Os recursos em questão são as horas dos desenvolvedores no processo. A abordagem utilizada foi o integrar o time de desenvolvimento no processo através do Planning Poker.

Estimando a complexidade dos requisitos funcionais

Planning Poker é uma técnica do Agile de estimativa de complexidade baseada em sequência de distribuição natural, a sequência de Fibonacci, e no estímulo a análise sem influência direta da estimativa dos demais desenvolvedores.

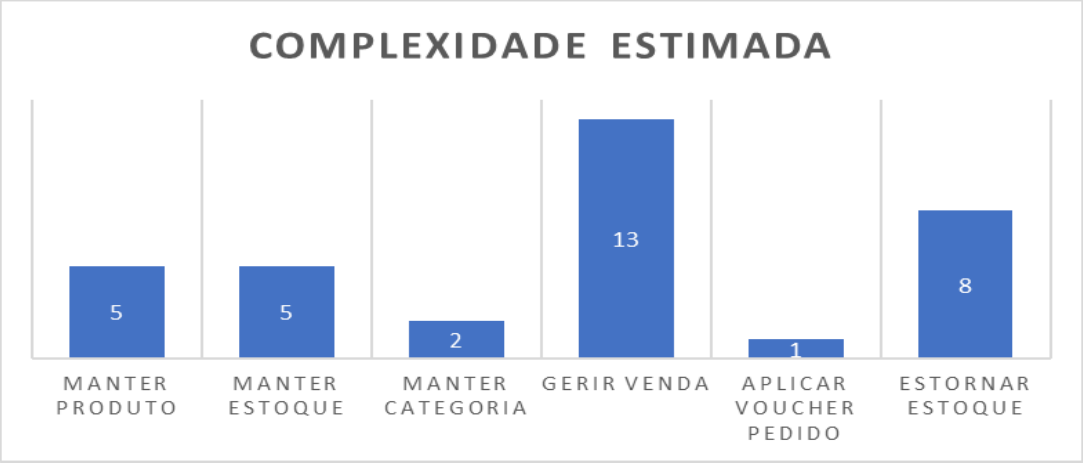
Após definida uma relação complexidade-número inicial, os demais requisitos são estimados em comparação aos demais. Não exigindo, nessa fase, valor de horas de desenvolvimento. O que está sendo avaliado é o grau de complexidade da tarefa.

O processo se dá da seguinte forma, dado um requisito funcional, cada desenvolvedor, escolhe uma carta contendo algum nos números iniciais da sequência de Fibonacci, de acordo com a sua estimativa pessoal da complexidade do requisito a ser avaliado.

Em seguida deverá colocar a carta virada para baixo na sua frente. Quando o todos tiverem feito o mesmo, todas as cartas são reveladas. Os participantes então, discutem e negociam a complexidade baseado na explicação do porquê colocou a carta indicada. Várias rodadas podem ser feitas até chegarem a um consenso. Se não houver, pode-se fazer uma média.

Depois do processo realizado temos uma tabela com as estimativas de complexidade de cada um dos requisitos funcionais.

Código	Requisito Funcional	Complexidade Estimada
[RF-001]	Manter Produto	5
[RF-002]	Manter Estoque	5
[RF-003]	Manter Categoria	2
[RF-004]	Gerir venda	13
[RF-005]	Aplicar voucher pedido	1
[RF-006]	Estornar estoque	8



Estimando tempo de desenvolvimento

Essa pontuação dos requisitos, é usada como medida para definir a velocidade de um time pelo padrão de entrega de histórias de usuários entregues a cada sprint.

Exemplo: O time A consegue entregar 22 pontos de complexidade por sprint de 2 semanas.

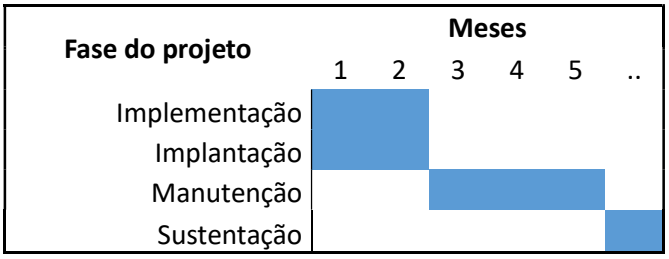
Por isso, é fundamental que haja um histórico de sprints entregues para que se chegue a um número mais preciso a partir da média das últimas iterações.

Observação: Para análise a fins acadêmicos não podemos usar a velocidade do time porque não temos histórico de desenvolvimento juntos. Portanto, a velocidade foi baseada nos *commits* de implementação real desse projeto em questão no GitHub.

Tem-se que a velocidade do time equiparasse a *1 ponto* de complexidade entregue a cada *8 horas* de desenvolvimento. Totalizando assim, *272 horas* de desenvolvimento para a entrega de cada um dos *34 pontos* de complexidade em requisitos de sistema.

Convertendo para dias, o projeto levará 1 mês e 17 dias de desenvolvimento. Esse tempo será acrescido de 10 dias de tarefas de suporte ao desenvolvimento como reuniões, definições arquiteturais e levantamento de requisitos. Totalizando 1 mês e 27 dias de projeto com atualizações a cada 2 semanas a partir da 2ª semana.

Cronograma



O aluno apresentou métrica de estimativa de custo para o desenvolvimento do sistema

Estimativa do custo do projeto

O custo de desenvolvimento é influenciado pelo tempo dos desenvolvedores de software no momento de implementação, implantação e manutenção.

Implementação

Definido o custo homem-hora dos profissionais envolvidos, baseado na média de salários devidamente acrescidos de encargos da CLT, temos o valor de R$ 56,82/hora.

Dado que a fase de Implementação tem 272 horas, o custo total dessa fase é estimado em R$ 15.454,54.

Contratualmente, o cliente pode optar por um serviço adicional de manutenção de software no qual esse poderá solicitar alterações nas funcionalidades de acordo com as horas extras contratadas pós implementação.

Disponíveis em 22 horas de desenvolvimento mensais para pequenos ajustes e correção de eventuais bugs. Novas funcionalidades não estão inclusas e não ser por novo contrato.

Nos 3 primeiros meses após a implantação a manutenção é obrigatória. Adicionando um custo de R$ 3750 para colocar a equipe a disposição da sustentação do sistema.

O contrato prevê possibilidade de estender o período de manutenção por tempo indeterminado pelo preço de R$1250/mês corrigidos pela inflação em custo de manutenção + custos de cloud proporcionais a quantidade de uso de recursos.

Totalizando assim, **R$**

30.246,86 + R$ 1.250,00/mês pós-implantação

Manutenção	R$ 3.750,00
Implantação	R$ 6.000,00
Implementação	R$ 15.454,55
Gerenciamento	20%
Margem de lucro	40%
CUSTO TOTAL	**R$ 30.246,86**

Manutenção extra	R$ 1.250,00

Primeira proposta de implantação

O aluno apresentou Implantação

Com a finalidade de cumprir os requisitos não-funcionais, a implantação do software não pode se dar em um ambiente de infraestrutura *on-premisse* dado que não seria possível cumprir o requisito de escalabilidade e disponibilidade sem um aumento muito significativo nos custos de manutenção e implantação.

A proposta sugerida de implantação faz necessária um ambiente em *cloud*, recomentando o Azure da Microsoft com os seguintes serviços:

- AKS (Serviço de Kubernetes do Azure)
- Registro de containers do Azure
- Aplicativo Web para Contêiners
- Banco de dados SQL do Azure
- Cache do Redis para Azure
- Armazenamento Blobs do Azure
- Key Vault
- API Gateway
- Firewall

Tal implementação terá um custo de configuração da equipe envolvida juntamente com o preço cobrado pelo mantenimento da cloud que é de R$ 6.000 nessa fase.

Cronograma de Releases