**PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS**

**PUC Minas Virtual**

**Pós-graduação *Lato Sensu* em Arquitetura de *Software* Distribuído**

Projeto Integrado

Relatório Técnico

Sistema de Gestão de Vida Animal

Marcio Gadelha

Porto, Portugal

Junho, 2022.

# Projeto Integrado – Arquitetura de Software Distribuído

**Sumário**

[Projeto Integrado – Arquitetura de Software Distribuído 2](#_Toc119609215)

[1. Introdução 3](#_Toc119609216)

[2. Cronograma do Trabalho 6](#_Toc119609217)

[3. Especificação Arquitetural da solução 7](#_Toc119609218)

[3.1 Restrições Arquiteturais 7](#_Toc119609219)

[3.2 Requisitos Funcionais 8](#_Toc119609220)

[3.3 Requisitos Não Funcionais 9](#_Toc119609221)

[3.4 Mecanismos Arquiteturais 10](#_Toc119609222)

[4. Modelagem Arquitetural 11](#_Toc119609223)

[4.1 Diagrama de Contexto 11](#_Toc119609224)

[4.2 Diagrama de Container 12](#_Toc119609225)

[4.3 Diagrama de Componentes 13](#_Toc119609226)

[4.4 Projeto das Interfaces 14](#_Toc119609227)

[4.5 Código da Aplicação 17](#_Toc119609228)

[5. Avaliação da Arquitetura (ATAM) 18](#_Toc119609229)

[5.1. Análise das abordagens arquiteturais 18](#_Toc119609230)

[5.2. Cenários 18](#_Toc119609231)

[Referências 20](#_Toc119609232)

## Introdução

Nos últimos anos, há uma grande tendência na sociedade como um todo em adquirir animais de estimação. Segundo pesquisa do Insurance Information Institute, por exemplo, a porcentagem da população que possui um pet saltou de 56% em 1988 para 67% em 2020. Com isso, a relação entre os pets e seus donos está cada dia mais próxima, sendo que muitos donos os consideram como membros da família.

E, assim como as pessoas querem o melhor para seus filhos, esses pais de pets querem o melhor para seus animais. O resultado é uma abundância de produtos e serviços que até pouco tempo eram desconhecidos.

Consequentemente, ocorreu um aumento na procura produtos destinados a pets, como uma alimentação mais adequada, suprimentos, brinquedos, roupas, higiene e consultas veterinárias. Além disso, surgiram novos serviços, como hotéis e até mesmo spa para os animais.

Neste sentido, o mercado está demonstrando ser promissor, pois apresentou um grande crescimento nos últimos anos. Em 2006, os americanos gastaram US$ 38,5 bilhões com seus animais de estimação. Já em 2018, eles quase dobraram esse número, para US$ 72,6 bilhões. Ou seja, o mercado praticamente dobrou em pouco mais de uma década. A maior parcela de vendas do mercado de animais domésticos está relacionada à alimentação. Nos EUA em 2018 foram gastos US$ 30,3 bilhões com alimentos pet e estimou-se um aumento de 4,3% para até o final de 2019. No Brasil esta projeção também é observada, a Associação Brasileira da Indústria de Produtos para Animais de Estimação (Abinpet) indicou que a alimentação pet ocupa 70% das vendas totais nesta área. Inclusive, a venda de alimentos industrializados para pets aumentou 28,5% no último ano. Ou seja, mesmo com novas opções o mercado ainda está em alta. Vale mencionar, que uma alimentação adequada é um dos principais fatores que contribuem para saúde dos animais. Por isso, a busca dos donos por adquirir alimentos de qualidade e mais sofisticados, resulta em uma grande potencialidade expansão do mercado pet.

Atualmente, o mercado pet apresenta um grande potencial para investimentos. A busca dos donos por mais serviços e produtos para animais de estimação, proporciona novas oportunidades de negócio. A divisão do Departamento de Trabalho nos EUA projeta um crescimento na área de animais domésticos de 16% entre 2018 e 2028, o que é bastante significativo. Em alguns lares, inclusive, existem mais animais de estimação do que crianças. Por que isso acontece? Isso está relacionado a mudanças culturais, uma vez que atualmente as pessoas estão esperando mais tempo para terem filhos. Entretanto, há necessidade de uma companhia, por isso, adotam um animal de estimação. Além disso, são observadas mudanças também na forma de tratamento dos pets, devido à maior proximidade entre os animais e seus donos, demonstrando um aumento da humanização de animais de estimação. Com isso, donos aumentaram a procura por produtos de alta qualidade e acessórios mais caros e gourmetizados. Devido à demanda, o mercado mantém-se em alta e ao mesmo tempo saturado quando falamos das opções tradicionais, como pet shops e clínicas veterinárias. Porém, existe um espaço enorme para opções inovadoras, como: restaurantes para pets, roupas humanizadas, hotéis, escola e aplicativos para encontrar cuidadores. Além de uma atenção especial com a alimentação, que envolve desde a cautela na escolha de ingredientes que compõem o pet food até priorizar alimentos que atendem exigências específicas para gatos e cachorros. Um grande diferencial desse segmento é a constante demonstração de ser consistente e anti crise, com alta de 13,5% em ano de pandemia, o setor pet vislumbrava grande crescimento em 2021. Em 2020, ano em que a economia brasileira encolheu pelo menos 4% e diferentes atividades econômicas amargaram perdas expressivas, o setor pet teve um crescimento estimado de 13,5% em relação a 2019, com faturamento acima dos R$ 40 bilhões, de acordo o Instituto Pet.

Mesmo diante desse enorme mercado consumidor, existe uma escassez de ferramentas de qualidade para suportar a administração e controle, bem como prover produtos e serviços de forma prática e organizada.

Em nossa versão inicial (MVP), temos como missão prover uma plataforma que proporcione os seguintes recursos:

* Administração de informações das características dos animais como peso, tamanho, vacinas, bem como fotos e vídeos;
* Ser integrada com um mecanismo de autenticação aberto e popular para facilitar o acesso de novos usuários;
* De fácil entendimento e com excelente usabilidade para atingir os mais diversos perfis de usuários;
* Ser robusta, escalável e tolerante a falhas para suportar o crescimento da operação a nível mundial;

Portanto, neste documento serão apresentados os requisitos arquiteturais, funcionais, não funcionais e as diagramações da solução para o desenvolvimento da plataforma 4Pet que compreenda os objetivos citados acima.

## Cronograma do Trabalho

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Datas** | | **Atividade / Tarefa** | **Produto / Resultado** |
| **De** | **Até** |
| 01/06/2022 | 01/06/2022 | 1. Cronograma do Trabalho | Construção desta tabela |
| 02/06/2022 | 02/06/2022 | 1. Contextualização do trabalho | Construção da contextualização deste projeto |
| 03/06/2022 | 05/06/2022 | 1. Definição dos requisitos Arquiteturais | Lista dos requisitos Arquiteturais identificados |
| 03/06/2022 | 05/06/2022 | 1. Definição dos requisitos Funcionais | Lista dos requisitos funcionais identificados |
| 03/06/2022 | 05/06/2022 | 1. Definição dos requisitos Não-funcionais | Lista dos requisitos Não-funcionais identificados |
| 03/06/2022 | 05/06/2022 | 1. Definição dos Mecanismos Arquiteturais | Lista dos Mecanismos Arquiteturais identificados |
| 06/06/2022 | 08/06/2022 | 1. Construção dos Diagramas de Contextos – Modelo C4 | Diagrama de contexto criado no Draw.io e documentado |
| 08/06/2022 | 15/06/2022 | 1. Revisão da Etapa 1 | Documento Etapa 1 revisado |
| 16/06/2022 | 20/06/2022 | 1. Construção do vídeo de apresentação da Etapa 1 | Vídeo criado da Etapa 1 |
| 10/08/2022 | 10/08/2022 | 1. Apresentação em PPT da Etapa 1 | PPT |
| 10/08/2022 | 10/08/2022 | 1. Publicação no repositório Github Etapa 1 | Arquivos produzidos no Github disponíveis abertamente |
| 11/08/2022 | 12/08/2022 | 1. Construção dos Diagramas de Contêineres | Diagramas de contêineres |
| 13/08/2022 | 15/08/2022 | 1. Construção dos Diagramas de Componentes | Diagramas de componentes |
| 16/08/2022 | 18/08/2022 | 1. Desenho dos Wireframes da POC | Protótipos de telas de baixa fidelidade |
| 19/08/2022 | 30/09/2022 | 1. Código da aplicação | Aplicação com 3 requisitos implementados |
| 30/09/2022 | 30/09/2022 | 16. Publicação no repositório Github Etapa 2 | Arquivos produzidos no Github disponíveis abertamente |
| 01/10/2022 | 03/10/2022 | 17. Análise das abordagens arquiteturais | Seção do documento produzido |
| 04/10/2022 | 05/10/2022 | 18. Cenários | Seção do documento produzido |
| 06/10/2022 | 07/10/2022 | 19. Evidências da avaliação | Seção do documento produzido |
| 08/10/2022 | 09/10/2022 | 20. Resultados obtidos | Seção do documento produzido |
| 10/10/2022 | 11/10/2022 | 21. Avaliação crítica dos resultados | Seção do documento produzido |
| 12/10/2022 | 13/10/2022 | 22. Conclusão | Seção do documento produzido |
| 14/10/2022 | 19/10/2022 | 23. Construção do vídeo de apresentação da Etapa 3 | Vídeo da etapa 3 disponível |
| 20/10/2022 | 20/10/2022 | 24. Publicação no repositório Github Etapa 3 | Arquivos produzidos no Github disponíveis abertamente |

## 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 macroarquitetura da solução.

## Restrições Arquiteturais

Os Requisitos Arquiteturais são todos os requisitos, sejam eles Funcionais ou Não-Funcionais que têm **impacto direto** sobre a Arquitetura do Sistema. Dessa forma, o Arquiteto precisa analisar os requisitos do sistema identificando algumas propriedades e então “filtrando” os Requisitos Arquiteturais. A lista a seguir apresenta os requisitos arquiteturais que foram identificados para implementação inicial da plataforma.

|  |  |
| --- | --- |
| **ID** | **Descrição** |
| RA01 | Deve ser usado tecnologias abertas (Open source) para o desenvolvimento de toda a plataforma. |
| RA02 | Deve ser usado o serviço de nuvem da Amazon Web Services como provedora da infraestrutura necessária para a plataforma. |
| RA03 | Deve ser usado o serviço OAuth 2.0 do Google bem como a possibilidade de criação de conta diretamente na plataforma, para o gerenciamento de autenticação dos usuários. |
| RA04 | A solução móvel (Apps Mobile) deve suportar os sistemas operacionais móveis mais populares, Android e IOS. |
| RA05 | Deve se implementar uma API RESTful para prover todos os dados e comunicação com os clientes de forma agnóstica e desacoplada com o backend, facilitando uma possível mudança tecnológica no frontend. |

## Requisitos Funcionais

Os Requisitos Funcionais são todos aqueles que estão associados às funcionalidades que ditam **o que** sistema deve fazer. A lista a seguir apresenta os requisitos funcionais identificados para o desenvolvimento inicial da plataforma.

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Descrição Resumida** | **Dificuldade (B/M/A)\*** | **Prioridade**  **(B/M/A)\*** |
| RF01 | O aplicativo deve permitir o cadastro de usuário através de uma conta do Google. | B | A |
| RF02 | O aplicativo deve permitir o cadastro de usuário através de um e-mail e senha. | B | A |
| RF03 | O aplicativo deve permitir a atualização do cadastro do usuário, podendo adicionar: endereço, coordenadas e telefone. | M | A |
| RF04 | O aplicativo deve permitir o cadastro de um pet, informando: Foto do Perfil, Tipo, Porte, Raça, Data de nascimento, Disponibilidade para cruzamento, Mãe de leite, Disponibilidade para adoção. | A | A |
| RF05 | O aplicativo deve permitir o cadastro do peso do pet e sua data de referência. | M | A |
| RF06 | O aplicativo deve permitir o cadastro de medidas do pet e sua data de referência. | M | A |
| RF07 | O aplicativo deve permitir o cadastro de medicamentos em uso pelo pet e sua periodicidade. | M | A |
| RF08 | O aplicativo deve permitir o cadastro de vacinas tomadas pelo pet e sua data de referência. | M | A |
| RF09 | O aplicativo deve permitir o cadastro de fotos do pet, com legenda e data referência. | A | A |
| RF10 | O aplicativo deve permitir o cadastro de vídeos do pet, com legenda e data referência. | A | A |
| RF11 | O aplicativo deve permitir a atualização de um pet, informando: Peso, Medidas, Medicamentos, Vacinas, Fotos e Vídeos. | A | A |
| RF12 | O aplicativo deve permitir os prestadores a cadastrar e disponibilizar seus serviços. | M | B |
| RF13 | O aplicativo deve permitir os vendedores a cadastrar e disponibilizar seus produtos. | M | B |
| RF14 | O aplicativo deve permitir os donos a pesquisar e adquirir produtos ou serviços. | A | B |
| RF15 | O aplicativo deve permitir os prestadores a consultar um relatório com estatísticas sobre seus serviços. | M | B |
| RF16 | O aplicativo deve permitir os vendedores a consultar um relatório com estatísticas sobre seus produtos. | M | B |
| RF17 | O aplicativo deve permitir os donos configurarem quais notificações e frequência deseja receber. | M | B |
| RF18 | O aplicativo deve permitir o dono avaliar o produto ou serviço adquirido. | M | B |
| RF19 | O aplicativo deve permitir o prestador de serviço configurar as notificações que deseja em relação a seus serviços disponíveis. | M | B |
| RF20 | O aplicativo deve permitir o vendedor configurar as notificações que deseja em relação a seus produtos disponíveis. | M | B |
| RF21 | O sistema deve enviar informações referentes aos serviços para o prestador respeitando suas configurações. | M | B |
| RF22 | O sistema deve enviar informações referentes aos produtos para o vendedor respeitando suas configurações. | M | B |

\*B=Baixa, M=Média, A=Alta.

## Requisitos Não Funcionais

Os Requisitos Não Funcionais estão associados às restrições de funcionalidades que **ditam como** o sistema deve fazer. A lista a seguir apresenta os requisitos não funcionais identificados para o desenvolvimento inicial da plataforma.

|  |  |  |
| --- | --- | --- |
| **ID** | **Descrição** | **Prioridade**  **B/M/A** |
| RNF01 | A plataforma deve habilitar a autenticação baseado no modelo OAuth2 da Google e diretamente no sistema. | A |
| RNF02 | O sistema deve operar em tempo integral em 24h x 7d, tendo disponibilidade mínima de 90%, e para atender essa necessidade deverá usar a hospedagem AWS com mecanismo de escala automática para responder ao aumento de demanda. | A |
| RNF03 | A comunicação entre o sistema Backend e Mobile da plataforma deve ser implementado através do padrão de serviços REST. | A |
| RNF04 | O sistema deverá utilizar para persistência um banco de dados NoSQL para atender a necessidade de alta performance na resposta em alta demanda, aceitando em contra partida uma integridade eventual. | A |
| RNF05 | As notificações por e-mail ou push devem operar por meio de filas de mensagens por não haver a necessidade de serem em tempo real e atenderem a grande demanda. | A |
| RNF06 | A versão Web da plataforma deve suportar os navegadores modernos, prioritariamente Google Chrome e Mozila Firefox. | M |

## Mecanismos Arquiteturais

Os mecanismos arquiteturais representam conceitos técnicos fundamentais que serão padronizados por toda a solução. Eles são refinados durante o projeto em três estados, representados pelas três categorias de Mecanismos Arquiteturais:

* Mecanismo de Análise, que dá ao mecanismo um nome, uma descrição resumida e alguns atributos básicos derivados dos requisitos do projeto.
* Mecanismo de Design, que são mais concretos e assumem alguns detalhes do ambiente de implementação.
* Mecanismo de Implementação, que especifica a exata implementação de cada mecanismo.

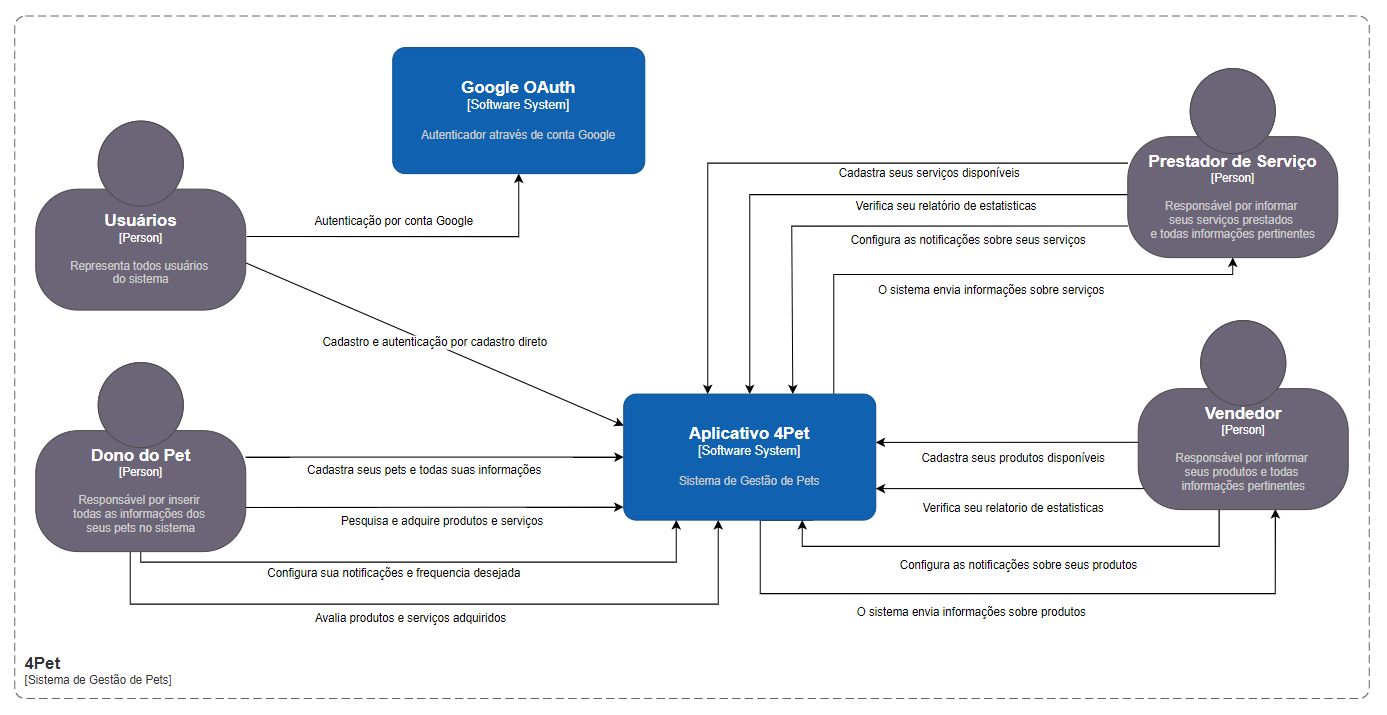
|  |  |  |
| --- | --- | --- |
| **Análise** | ***Design*** | **Implementação** |
| Persistência | ORM | JPA / Hibernate / Spring Data |
| Persistência | Banco de Dados NoSQL | Amazon DynamoDB |
| Front end | GraphQL | Spring GraphQL |
| Front end | Mobile | Google Flutter |
| Front end | Navegador Web | Mozilla Firefox / Google Chrome |
| Teste de Software | Testes Unitários | JUnit |
| Usabilidade | Notificação por Push | Amazon Pinpoint |
| Autenticação | OAuth2 | Google Open Authorization 2 |

## Modelagem Arquitetural

Desejamos criar uma solução focada no usuário que envolva mobilidade, visando ser uma ferramenta prática que esteja sempre nas mãos de donos de animais de estimação para ajudar dos cuidados diários com seus pets, bem como registrar seus momentos e adquirir produtos e serviços. Para isso através do diagrama de contexto abaixo demonstramos a criação de um componente de backend responsável por todos os serviços no servidor com regras de negócio e tendo um cliente leve apenas focado na apresentação baseado em uma solução hibrida utilizando o Ionic.

Para esta modelagem arquitetural optou-se por utilizar o modelo C4 para documentação de arquitetura de software. Mais informações a respeito podem ser encontradas aqui: <https://c4model.com/> e aqui: <https://www.infoq.com/br/articles/C4-architecture-model/>. Dos quatro níveis que compõem o modelo C4 três serão apresentados aqui e somente o Código será apresentado na próxima seção (5).

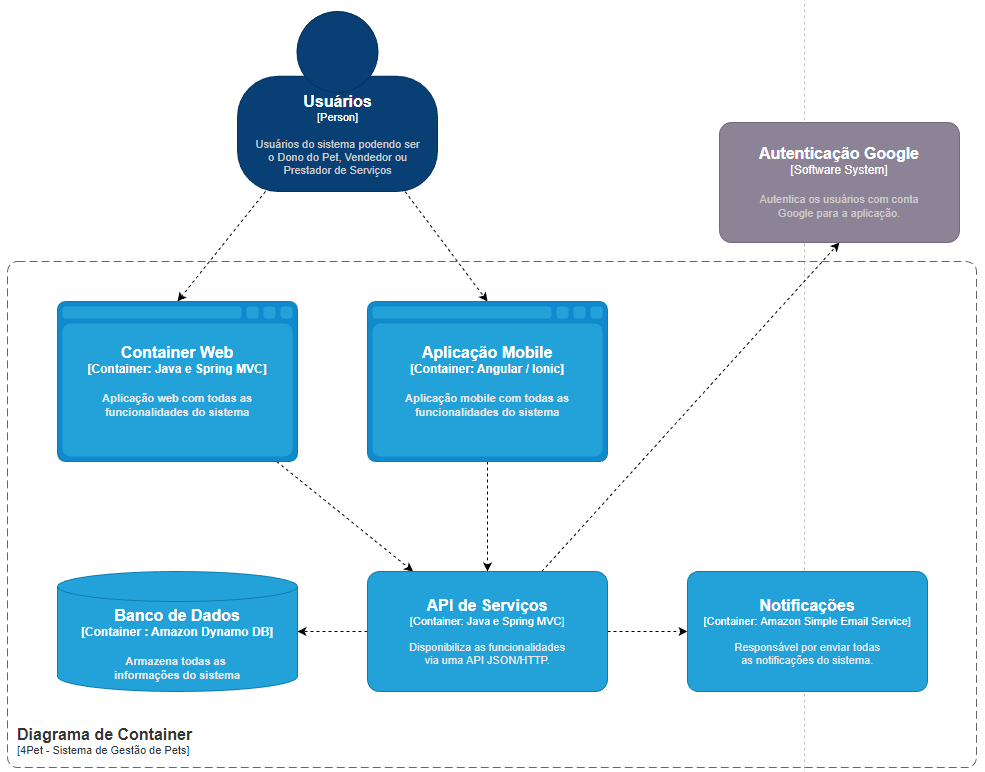
## 4.1 Diagrama de Contexto



**Figura 1 - Visão Geral da Solução 4Pet.**

A figura 1 mostra a especificação o diagrama geral da solução proposta, com todos seus principais sistemas e pessoas envolvidas nos processos de gestão de vida animal.

## 4.2 Diagrama de Container



**Figura 2 – Diagrama de Container da Solução 4Pet.**

A figura 2 mostra o diagrama de container no padrão C4 model para a solução proposta, com todos seus principais containers envolvidos.

## 4.3 Diagrama de Componentes

## 

**Figura 3 – Diagrama de Componentes da Solução 4Pet.**

A figura 3 mostra o diagrama de componentes da solução proposta, com todos seus principais componentes, estes podem ser detalhados como:

* Angular / Ionic – Plataforma/Framework utilizado para o desenvolvimento da aplicação mobile.
* Java / Spring MVC – Plataforma/Framework utilizado para o desenvolvimento da API que serve como o backend de todo sistema.
* Container Web – Aplicação web que proverá alguns recursos do sistema através do navegador.
* Usuario Controller – Provê os serviços de acesso para todos os usuários.
* Produto Controller – Provê os serviços relacionados a produtos para Pet.
* Servico Controller – Provê os serviços relacionados aos serviços para Pet.
* Pet Controller – Provê os serviços relacionados a gestão do Pet.
* Relatorio Controller – Provê os serviços relacionados a geração de relatórios do Pet.
* Notificacao Controller – Provê os serviços para gerenciar as notificações dos usuários.
* Google OAuth – Provê os serviços de acesso através de uma conta Google.
* Banco de Dados – Componente de persistência que provê todos os serviços para gerenciar os dados de todo sistema.
* Notificações – Componente JMS que executa de forma assíncrona as notificações dos usuários do sistema.

## 4.4 Projeto das Interfaces

Neste capítulo demonstramos as principais interfaces, através dos wireframe e o protótipo navegável da solução.

Wireframe: <https://bit.ly/3yISkcS>

Protótipo: <https://bit.ly/3Ep1O0e>.



**Figura 4 – Interface inicial para Login.**



**Figura 5 – Interface inicial para Cadastro de Usuário.**

**

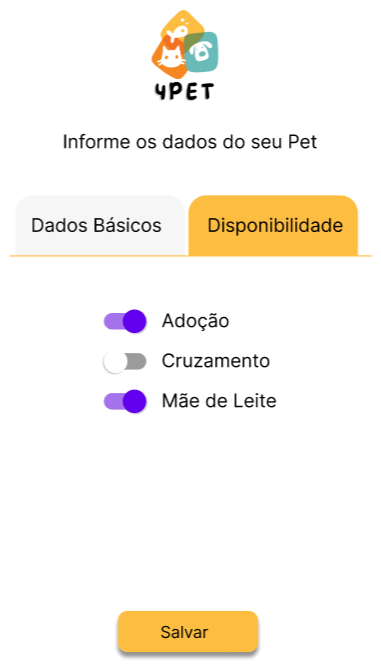
**Figura 6 – Interface inicial da aplicação após a autenticação (Home).**



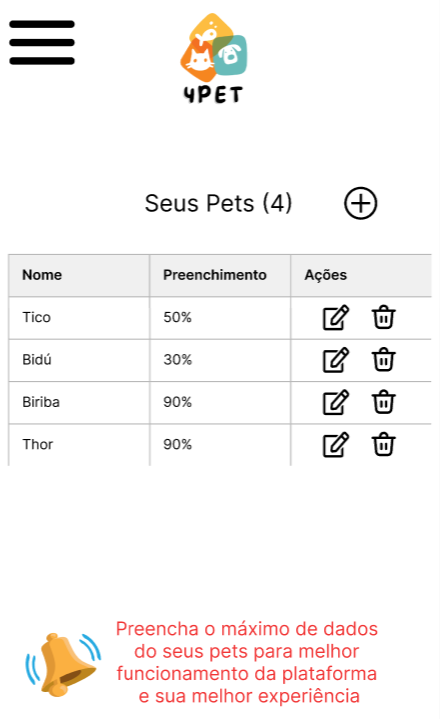
**Figura 7 – Interface do menu principal da aplicação.**



**Figura 8 – Interface do cadastro de Pet.**



**Figura 9 – Interface de configuração de disponibilidades do Pet.**



**Figura 10 – Interface inicial com Pets cadastrados (Home).**

## 4.5 Código da Aplicação

Neste capítulo demonstramos em detalhes como foi desenvolvido a aplicação.

Foram criados dois macro componentes arquiteturais, sendo um frontend e outro backend.

O frontend foi criado coma tecnologia híbrida baseada em Ionic e Angular, com isso, sendo necessário criar um único código fonte gerar os aplicativos para ambos os sistemas móveis.

O backend foi criado baseado em Java 8 e Spring Boot, com micro serviços, visado ter a capacidade de uma escala horizontal, redundância a alta disponibilidade.

**Repositórios:**

Frontend: <https://github.com/zagadelha/4-pet>

Branch: master

Backend: <https://github.com/zagadelha/tcc-mba-puc>

Branch: aws-dynamo-db

Diretório: tcc-mba-puc/code/4pet-backend/

## Avaliação da Arquitetura (ATAM)

A avaliação da arquitetura desenvolvida neste trabalho é abordada nesta seção visando avaliar se ela atende aos requisitos segundo o método ATAM.

## 5.1. Análise das abordagens arquiteturais

|  |  |  |  |
| --- | --- | --- | --- |
| **Atributos de Qualidade** | **Cenários** | **Importância** | **Complexidade** |
| Segurança | Cenário 1: A plataforma deve habilitar a autenticação baseado no modelo OAuth2 da Google e diretamente no sistema. | M | M |
| Disponibilidade | Cenário 2: O sistema deve estar disponível 24x7x365, tendo disponibilidade mínima de 90%. | M | M |
| Eficiência | Cenário 3: A comunicação entre o sistema Backend e Mobile da plataforma deve ser implementado através do padrão de serviços REST. | A | A |
| Desempenho | Cenário 4: O sistema deverá utilizar para persistência um banco de dados NoSQL para atender a necessidade de alta performance na resposta em alta demanda, aceitando em contra partida uma integridade eventual. | A | M |
| Desempenho | Cenário 5: As notificações por e-mail ou push devem operar por meio de filas de mensagens por não haver a necessidade de serem em tempo real e atenderem a grande demanda. | M | M |
| Compatibilidade | Cenário 6: A versão Web da plataforma deve suportar os navegadores modernos, prioritariamente Google Chrome e Mozila Firefox. | M | M |

## 5.2. Cenários

Cenário 1 – Segurança: Para efetuar seu cadastro, o usuário terá a opção de utilizar sua conta do Google ou efetuar o cadastro diretamente no sistema informando um email e senha desejados.

Cenário 2 – Disponibilidade: As funcionalidades do sistema devem manter uma disponibidade de no mínimo 90%, 24 horas por dia, 7 dias por semana, suportando uma carga planejada de milhares de usuários com acesso simultâneo ao aplicativo.

Cenário 3 – Eficiência: Com a alta carga de milhares de acessos simultâneos a comunicação da API escolhida foi no padrão REST por ser menos verbosa e mais eficiente melhorando o tempo de resposta e dados trafegados através da rede.

Cenário 4 – Desempenho: Com a alta carga de milhares de acessos simultâneos a persistência escolhida foi a NoSQL visando uma alta velocidade de resposta, aceitando em contra partida uma integridade eventual.

Cenário 5 – Desempenho: Com o planejamento de alta demanda de usuários e configurações de alertas, foi utilizado um serviço de mensageria para gerenciar essa demanda de mensagens e enviar sob demanda, pois não existe a necessidade de envio em tempo real.

Cenário 6 – Compatibilidade: O front-end deve se manter funcional e preservar o design sendo executado nos navegadores Chrome e Firefox.

## 6.3. Evidências da Avaliação

|  |  |
| --- | --- |
| Atributo de Qualidade: | Segurança |
| Requisito de Qualidade: | A plataforma deve habilitar a autenticação baseado no modelo OAuth2 da Google e diretamente no sistema. |
| Preocupação: | |
| O sistema deve permitir a identificação do responsável pelas alterações no cadastro de clientes, além disso deve registrar quais informações foram alteradas. | |
| Cenário(s): | |
| Cenário 1 | |
| Ambiente: | |
| Sistema em operação normal | |
| Estímulo: | |
| Sistemas enviam requisições para a API rest do sistema My-Customers | |
| Mecanismo: | |
| Elaborar inteligência que receba as requisições que mudem o estado do cadastro do cliente e envie para o sistema responsável por persistir os logs. | |
| Medida de resposta: | |
| Persistir os logs enviados e possibilitar consulta através de interface para usuário. | |
| Considerações sobre a arquitetura: | |
| Riscos: | Instabilidade no servidor responsável por armazenamento dos logs causando perda de pacotes. |
| Pontos de Sensibilidade: | Não há |
| Tradeoff: | Não há |

|  |  |
| --- | --- |
| Atributo de Qualidade: | Performance |
| Requisito de Qualidade: | O sistema deve atender a no mínimo 340 tps. |
| Preocupação: | |
| O sistema deve atender a um número mínimo de requisições por segundo pois terá de atender a um cenário onde milhares de usuários farão sua utilização simultaneamente. | |
| Cenário(s): | |
| Cenário 2 | |
| Ambiente: | |
| Sistema em operação extrema | |
| Estímulo: | |
| Teste de carga no sistema. Não é possível reproduzir pois o hardware disponível para testes não consegue atingir a métrica definida. | |
| Mecanismo: | |
| A infraestrutura que suporta o sistema deve ser escalável a fim de atender a métrica definida caso necessário. | |
| Medida de resposta: | |
| Escalar a infraestrutura de forma que o tempo de resposta não aumente diante ao aumento no número de requisições. | |
| Considerações sobre a arquitetura: | |
| Riscos: | Infraestrutura não suportar o aumento no número de requisições e aumentar o tempo de resposta para os usuários. |
| Pontos de Sensibilidade: | Não há |
| Tradeoff: | Não há |

|  |  |
| --- | --- |
| Atributo de Qualidade: | Disponibilidade |
| Requisito de Qualidade: | O sistema deve estar disponível 24x7x365. |
| Preocupação: | |
| O sistema deve apresentar a maior disponibilidade possível já que atende a um modelo de negócio *b2c*. | |
| Cenário(s): | |
| Cenário 3 | |
| Ambiente: | |
| Sistema em operação normal | |
| Estímulo: | |
| Usuários utilizando o sistema | |
| Mecanismo: | |
| Acessar sistema em diferentes horários e dias. | |
| Medida de resposta: | |
| Implantar sistema em infraestrutura resiliente a falhas. Para implementação das provas de conceito foi utilizado a nuvem da AWS, em sua versão paga ela pode garantir até 99.9% de disponibilidade. | |
| Considerações sobre a arquitetura: | |
| Riscos: | Sistema ficar indisponível, impossibilitando clientes de acessarem seus recursos financeiros. |
| Pontos de Sensibilidade: | Não há |
| Tradeoff: | Não há |

|  |  |
| --- | --- |
| Atributo de Qualidade: | Compatibilidade |
| Requisito de Qualidade: | O front-end deve ser compatível com os navegadores: Chrome, Firefox e Edge. |
| Preocupação: | |
| O front-end deve se manter funcional e preservar o design sendo executado nos navegadores Chrome, Firefox e Edge. | |
| Cenário(s): | |
| Cenário 4 | |
| Ambiente: | |
| Sistema em operação normal. | |
| Estímulo: | |
| Usuário acessa o front-end em diferentes navegadores. | |
| Mecanismo: | |
| Adequar código do front-end para diferentes navegadores caso necessário. | |
| Medida de resposta: | |
| Design é apresentado de forma idêntica e sistema permanece funcional em diferentes navegadores.  Edge:  Chrome:  Firefox: | |
| Considerações sobre a arquitetura: | |
| Riscos: | O código do front-end pode ser interpretado de maneiras diferentes entre os navegadores, causando diferença entre o layout e até impedir que funcionalidades sejam executadas corretamente. |
| Pontos de Sensibilidade: | Não há |
| Tradeoff: | Não há |

|  |  |
| --- | --- |
| Atributo de Qualidade: | Usabilidade |
| Requisito de Qualidade: | O front-end deve apresentar características que facilitem a utilização por PCD’s. |
| Preocupação: | |
| O front-end deve fornecer boa usabilidade. | |
| Cenário(s): | |
| Cenário 5 | |
| Ambiente: | |
| Sistema em operação normal. | |
| Estímulo: | |
| Usuários utilizando o sistema. | |
| Mecanismo: | |
| Acompanhar utilização do sistema verificando tempo que tarefas levam para serem completadas. | |
| Medida de resposta: | |
| Sistema com design intuitivo e de fácil visualização.  Na prova de conceito implementada é possível efetuar a atualização de um cadastro com a execução de apenas dois comandos.  Foram implementados campos grandes que facilitam a visualização, o botão de ação foi destacado com uma cor forte que torna fácil sua identificação. | |
| Considerações sobre a arquitetura: | |
| Riscos: | Um sistema difícil de utilizar causa perda de produtividade e pode até mesmo impossibilitar a utilização de alguns usuários. |
| Pontos de Sensibilidade: | Não há |
| Tradeoff: | Não há |

|  |  |
| --- | --- |
| Atributo de Qualidade: | Manutenibilidade |
| Requisito de Qualidade: | Pipelines de deploy automático para todos os sistemas envolvidos na solução. |
| Preocupação: | |
| Mitigar erros com publicações manuais. | |
| Cenário(s): | |
| Cenário 6 | |
| Ambiente: | |
| Sistema em operação normal. | |
| Estímulo: | |
| Desenvolvedor faz publicação de versão em produção. | |
| Mecanismo: | |
| Pipeline faz todos os processos necessários para publicação e publica a versão em produção.  Para o desenvolvimento da POC não foram implementados pipelines para deploy automático, porém, como foi utilizado o Elastic Beanstalk é possível fazer o deploy automático utilizando o próprio Visual Studio. | |
| Medida de resposta: | |
| Versão é publicada automaticamente em produção. | |
| Considerações sobre a arquitetura: | |
| Riscos: | Algum agente de publicação falhar impossibilitando o deploy automático. |
| Pontos de Sensibilidade: | Não há |
| Tradeoff: | Não há |

|  |  |
| --- | --- |
| Atributo de Qualidade: | Eficiência |
| Requisito de Qualidade: | O processamento de grandes quantidades de informação não deve afetar a operação do sistema. |
| Preocupação: | |
| Sistema tornar-se lento devido a interferência de processamentos de grandes quantidades de informação. | |
| Cenário(s): | |
| Cenário 7 | |
| Ambiente: | |
| Sistema com grande carga de processamento. | |
| Estímulo: | |
| Processamentos que demandam grande quantidade de recursos sendo executados. | |
| Mecanismo: | |
| Executar processamento de dados que necessitam de maior quantidade de recursos fora do horário crítico de utilização do sistema. | |
| Medida de resposta: | |
| Em um horário com menor utilização do sistema a tendência é que ocorra pouca ou nenhuma interferência para os usuários. | |
| Considerações sobre a arquitetura: | |
| Riscos: | Algum processamento não terminar de executar antes do início do próximo dia. |
| Pontos de Sensibilidade: | Não há |
| Tradeoff: | Não há |

## 6.4. Resultados Obtidos

|  |  |  |
| --- | --- | --- |
| **Requisitos Não Funcionais** | **Teste** | **Homologação** |
| RNF01: Deve ser registrado trilha de auditoria para todas as alterações que ocorrerem no cadastro de clientes | N.A | N.A |
| RNF02: As API’s devem possuir um *throughput* de no mínimo 340 tps | N.A | N.A. |
| RNF03: O sistema deve possuir disponibilidade 24x7x365 | OK | OK |
| RNF04: O front-end deve ser compatível com os navegadores: Chrome, Firefox e Edge | OK | OK |
| RNF05: O front-end deve apresentar características de design que facilitem o uso por PCD’s | OK | OK |
| RNF06: O deploy em produção das aplicações e a infraestrutura devem ser automatizadas usando pipelines CI/CD | OK | OK |
| RNF07: O processamento de grandes quantidades de dados deve ser feito preferencialmente após as 20h | N.A | N.A |

## Avaliação Crítica dos Resultados

A arquitetura proposta mostrou-se eficaz em resolver os principais problemas encontrados em sistemas de gestão de clientes em bancos, a utilização da arquitetura orientada a eventos permitiu centralizar as informações de clientes, o que facilita o mapeamento de informações, além disso, com a utilização da *stack* ELK a rastreabilidade de alterações nessas informações é feita de forma eficaz e rápida.

Através da implementação das POCS foi possível validar que as decisões em relação a tecnologias foram assertivas, desde a utilização de Angular como linguagem para o front end, passando pela utilização do Dapper como ORM até chegar no design pattern da API, para o último ponto foi observado que apesar de facilitar a implementação de regras de negócio pela equipe técnica não é possível que a área usuária altere as regras dinamicamente.

Com a utilização de um API Gateway toda a estrutura de integrações com o My-Customers é mapeada, essa solução também impede que novas integrações sejam feitas de forma desconhecida, assim obtemos um ambiente conhecidos por todos o que facilita a manutenibilidade e identificação de problemas.

Diante da implementação do Open Banking no Brasil torna-se essencial pensar em soluções que atendam a essa demanda, não foram especificadas soluções de segurança robustas que permitam tranquilidade em expor as informações da instituição para as demais instituições participantes.

## Conclusão

Esse trabalho teve como objetivo apresentar a aplicação da arquitetura de software através da implementação de um sistema fictício para controle de clientes em bancos, o que se mostrou totalmente vantajoso.

O desenvolvimento do relatório técnico deixou claro a importância de projetar software antes de desenvolver, os tópicos abordados trouxeram à tona problemas e permitiram a discussão para solução antes que esses ocorressem, sem a análise arquitetural esses problemas possivelmente seriam identificados já com o sistema em produção.

Os diagramas construídos possibilitaram visualizar todo o ecossistema da solução, incluindo integrações com os demais sistemas internos e externos, esse ponto de vista facilitou a tomada decisões sobre onde investir em segurança, performance e usabilidade.

A materialização de soluções através das POC’s foi crucial para identificar a aderências das tecnologias escolhidas, ampliar o conhecimento técnico e obter visibilidade para evoluções no projeto.

A partir dessas percepções pretende-se desenvolver melhorias no projeto, como por exemplo a implementação de *feature* *toggles* para controle dinâmico de regras de negócio pela área usuária, além disso, é necessário implementar soluções mais robustas de segurança que permitam atender a demanda do Open Banking com tranquilidade.

## Referências

BRF INGREDIENTS. **Crescimento do mercado pet e seu papel na nutrição animal.**

Disponível em: <https://bit.ly/3OBxYIt>.

Acesso em: 24 de junho de 2022.

EXAME. **Mercado sem crise: com alta de 13,5% em ano de pandemia, o setor pet crescerá mais em 2021.**

Disponível em: <https://bit.ly/3u0Fsw4>.

Acesso em: 24 de junho de 2022.

CNN BRASIL. **Mercado pet deve ter crescimento de 14% em 2022, projeta instituto.**

Disponível em: <https://bit.ly/3ndS5jN>.

Acesso em: 24 de junho de 2022.

SEBRAE. **Mercado PET fatura quase 35 bi ao ano e tende a crescer.**

Disponível em: <https://bit.ly/3u3vhqO>.

Acesso em: 24 de junho de 2022.