

**Documentação de um**

**Produto de Software**

**BICHO EXPRESS**

**Nome dos Alunos:**

Maria Eduarda Medeiro Porto- 824144948

Matheus Alves Santana- 824144952

Erick Domingues Soares- 82414486

Wellington de Oliveira Sousa - 825240209

Matheus Henrique da Costa e Silva – 82410661

Eduardo Baptistella Gonçalves- 824147595

Gabriel Prieto Lima- 824142064

**2025**

**ÍNDICE DETALHADO**

**1.** **Introdução 3**

1.1. Tema 3

1.2. Objetivos a serem alcançados 3

1.3. Escopo principal 3

**2.** **Definição do Modelo de Processo 4**

**3.** **Requisitos do Sistema de Software 4**

3.1. Requisitos Funcionais 4

3.2. Requisitos Não-Funcionais 4

**4.** **Projeto 5**

4.1. Arquitetura Lógica 5

4.2. Arquitetura Física 5

**5.** **Protótipo de Interface 6**

**6.** **Critérios de Qualidade de Software 7**

**7. Testes 8**

7.1. Plano de Testes 8

7.2. Roteiro de Testes 8

**Anexo I 9**

1. **Introdução**
   1. **Tema**

O projeto visa o desenvolvimento de um sistema de transporte especializado para animais de estimação, que funcione como uma plataforma digital de mobilidade pet-friendly. A solução será composta por um aplicativo móvel para usuários (donos de pets) e motoristas, além de um portal administrativo para gestão do serviço. O sistema deverá contemplar funcionalidades específicas para o transporte seguro e confortável dos animais, como cadastro detalhado do pet, requisitos de segurança, monitoramento em tempo real e comunicação direta entre usuário e motorista.

A escolha de São Paulo como sede é estratégica, pois a cidade possui uma grande concentração de animais de estimação e uma demanda crescente por serviços especializados, além de desafios logísticos típicos de grandes metrópoles, como trânsito intenso e diversidade de perfis de usuários.

A arquitetura do sistema será baseada em microserviços para garantir escalabilidade e facilidade de manutenção, utilizando APIs RESTful para comunicação entre front-end e back-end. A interface será desenvolvida com foco em usabilidade e acessibilidade, seguindo padrões modernos de design responsivo

**Objetivos a serem alcançados**

O principal objetivo do projeto é criar uma plataforma digital intuitiva e confiável para o transporte de animais de estimação, que atenda às necessidades dos donos e garanta o bem-estar dos pets durante o trajeto. Especificamente, pretende-se:

* Desenvolver um aplicativo móvel multiplataforma (iOS e Android) e um portal web para cadastro, solicitação e gerenciamento de corridas.
* Implementar funcionalidades de geolocalização em tempo real, cálculo automático de rotas e estimativa de preços.
* Garantir a segurança dos animais com requisitos técnicos, como motoristas treinados, veículos adaptados e monitoramento via câmera.
* Automatizar processos de agendamento, pagamento e avaliação dos serviços.
* Oferecer suporte ao cliente integrado via chat e notificações push.
* Atender ao público-alvo composto por donos de animais de estimação residentes em São Paulo e região metropolitana, que buscam uma alternativa prática e segura para transportar seus pets.
* Promover inovação no mercado pet, integrando tecnologias de IoT para monitoramento do bem-estar animal durante o transporte.
  1. **Escopo principal**

A proposta principal do projeto é desenvolver um sistema completo de transporte de animais que englobe as seguintes ações e implementações:

* Cadastro e perfil do usuário e do pet: inclusão de informações detalhadas sobre o animal (raça, porte, necessidades especiais) e do dono.
* Solicitação e agendamento de corridas: interface para o usuário solicitar transporte imediato ou agendado, com opções de tipos de veículos adaptados.
* Geolocalização e roteirização: integração com APIs de mapas para rastreamento em tempo real e otimização de rotas.
* Sistema de pagamento integrado: suporte a múltiplas formas de pagamento digitais, com segurança e transparência.
* Avaliação e feedback: sistema de avaliação para motoristas e usuários, promovendo a qualidade do serviço.
* Painel administrativo: dashboard para gestão de motoristas, corridas, faturamento e suporte.
* Notificações e comunicação: envio de alertas via SMS, e-mail e push para status da corrida e comunicação direta entre partes.
* Segurança e conformidade: implementação de protocolos para garantir a segurança dos animais e conformidade com legislações locais.

O desenvolvimento seguirá metodologias ágeis, com entregas incrementais e foco na experiência do usuário, garantindo que o sistema seja escalável e adaptável às necessidades futuras.

1. **Definição do Modelo de Processo**

A empresa adotou o Scrum como modelo de desenvolvimento para a plataforma de transporte Bicho Express, aproveitando sua flexibilidade e agilidade para projetos complexos e dinâmicos. Com ciclos curtos de trabalho (sprints), estamos entregando funcionalidades rapidamente, permitindo ajustes contínuos baseados no feedback dos usuários e nas demandas do mercado.

Essa metodologia garante foco constante na experiência do usuário, priorizando usabilidade, segurança e inovação. Além disso, facilita a coordenação eficiente entre as equipes multidisciplinares, desde o desenvolvimento mobile até o suporte, assegurando que a arquitetura baseada em microserviços seja implementada de forma escalável e modular.

Com o Scrum, o projeto conta com total transparência no progresso, redução de riscos e entregas contínuas de valor. Isso acelera o lançamento no mercado, garantindo que a solução atenda às expectativas reais dos clientes e fortaleça a posição da empresa no mercado de mobilidade pet.

Optamos por não utilizar modelos tradicionais como Cascata, Iterativo, Prototipação ou Espiral devido às suas limitações em lidar com a complexidade, necessidade de adaptação rápida e constante evolução do projeto. Esses modelos são mais rígidos ou burocráticos, o que poderia dificultar a incorporação de feedbacks e atrasar entregas essenciais, enquanto o Scrum oferece a agilidade e colaboração necessárias para garantir o sucesso dessa iniciativa inovadora.

# 3. Requisitos do Sistema de Software

## 3.1. Requisitos Funcionais

Neste item devem ser descritos os requisitos a serem atendidos funcionalmente pelo sistema de uma forma simples, possibilitando a compreensão do comportamento do sistema pela perspectiva do usuário. Os requisitos funcionais podem ser representados por meio de modelo de caso de uso ou user stories (cartão de histórias).  
**Referência:** UC Modelos, métodos e técnicas da engenharia de software

### 3.1.1 Requisitos do Usuário (Tutor de Pets)

* **3.1.1.1** O sistema deve permitir cadastro/login via e-mail, senha, Google e autenticação multifator (MFA).
* **3.1.1.2** O usuário deve poder cadastrar múltiplos pets, incluindo espécie, porte, raça, peso, idade, vacinas e necessidades especiais.
* **3.1.1.3** O usuário deve poder solicitar transporte imediato ou agendado.
* **3.1.1.4** O sistema deve exibir o motorista mais próximo, estimativa de tempo e valor da corrida.
* **3.1.1.5** O usuário deve poder acompanhar o transporte em tempo real no mapa.
* **3.1.1.6** O sistema deve oferecer pagamentos via Pix, cartão de crédito/débito ou carteira digital.
* **3.1.1.7** O usuário deve poder cancelar corridas, respeitando regras de cancelamento.
* **3.1.1.8** O usuário deve poder avaliar motoristas após as corridas.
* **3.1.1.9** O usuário deve poder visualizar histórico de corridas e pagamentos.
* **3.1.1.10** O usuário deve poder acionar suporte via chat.

### 3.1.2 Requisitos do Motorista/Funcionário

* **3.1.2.1** O sistema deve permitir cadastro de motoristas, incluindo documentos obrigatórios (CNH, antecedentes, CRLV, seguro).
* **3.1.2.2** O motorista só poderá realizar corridas após validação de um gestor.
* **3.1.2.3** O motorista deve poder indicar status: **Disponível / Ocupado / Offline**.
* **3.1.2.4** O motorista deve receber solicitações de transporte e aceitá-las ou recusá-las.
* **3.1.2.5** O motorista deve visualizar informações do pet (porte, cuidados, restrições).
* **3.1.2.6** O motorista deve poder atualizar o status da corrida (a caminho, embarcado, em andamento, concluído).
* **3.1.2.7** O motorista deve poder visualizar histórico de corridas e ganhos.
* **3.1.2.8** O motorista deve receber notificações sobre corridas agendadas.
* **3.1.2.9** O motorista deve poder registrar ocorrências durante a corrida.

### 3.1.3 Requisitos do Setor Administrativo

* **3.1.3.1** O setor administrativo deve poder cadastrar, alterar ou excluir motoristas e usuários em situações de suporte.
* **3.1.3.2** O setor administrativo deve monitorar corridas em andamento por painel de controle.
* **3.1.3.3** O setor administrativo deve intervir em disputas entre usuários e motoristas.
* **3.1.3.4** O setor administrativo deve emitir relatórios detalhados de corridas, faturamento e cancelamentos.
* **3.1.3.5** O setor administrativo deve enviar mensagens de suporte via chat interno.
* **3.1.3.6** O setor administrativo deve acompanhar estatísticas de avaliação de motoristas.

### 3.1.4 Requisitos do Gestor

* **3.1.4.1** O gestor deve validar motoristas após análise documental.
* **3.1.4.2** O gestor deve suspender ou bloquear motoristas/usuários denunciados.
* **3.1.4.3** O gestor deve configurar tarifas dinâmicas (preço por km, porte do pet, taxa mínima).
* **3.1.4.4** O gestor deve criar campanhas promocionais e cupons de desconto.
* **3.1.4.5** O gestor deve acessar relatórios estratégicos (receita mensal, clientes ativos, cancelamentos).
* **3.1.4.6** O gestor deve acessar dashboards com previsões de demanda (região e horários de pico).

## 3.2. Requisitos Não Funcionais

Neste item devem ser apresentados os requisitos não funcionais, que especificam restrições sobre os serviços ou funções providas pelo sistema.

### 3.2.1 Requisitos Externos

#### 3.2.1.1 Interoperabilidade

* **3.2.1.1.1** O sistema deve permitir login integrado com Google.
* **3.2.1.1.3** O sistema deve integrar APIs bancárias, CPF, CEP e telefone para validações.

#### 3.2.1.2 Éticos

* **3.2.1.2.1** O sistema deve garantir transparência nos preços antes da corrida.
* **3.2.1.2.2** Apenas motoristas aprovados podem transportar pets.
* **3.2.1.2.3** O sistema deve respeitar princípios de bem-estar animal.

#### 3.2.1.3 Legais

* **3.2.1.3.1** O sistema deve estar em conformidade com a **LGPD**.
* **3.2.1.3.2** O sistema deve utilizar criptografia para proteger dados sensíveis.
* **3.2.1.3.3** O sistema deve seguir a norma **ISO/IEC 27001** para segurança da informação.

### 3.2.2 Requisitos Organizacionais

#### 3.2.2.1 Entrega

* **3.2.2.1.1** Apenas motoristas aprovados podem realizar corridas.
* **3.2.2.1.2** O sistema deve filtrar motoristas por localização.
* **3.2.2.1.3** O sistema permite pagamento somente por meios digitais.
* **3.2.2.1.4** Perfis denunciados e confirmados devem ser bloqueados.
* **3.2.2.1.5** O sistema deve permitir comunicação via chat criptografado.

#### 3.2.2.2 Implementação

* **3.2.2.2.1** Todas as páginas devem carregar em até 2 segundos.
* **3.2.2.2.2** Notificações devem ser entregues em até 5 segundos.
* **3.2.2.2.3** Campos de texto devem acionar teclado automaticamente.
* **3.2.2.2.4** O texto deve seguir fonte padrão, com títulos em negrito.

#### 3.2.2.3 Padrões

* **3.2.2.3.1** Todas as telas devem seguir design consistente.
* **3.2.2.3.2** Botões e campos devem ter comportamento previsível.
* **3.2.2.3.4** O código deve seguir boas práticas de desenvolvimento seguro.

### 3.2.3 Requisitos de Facilidade de Uso

* **3.2.3.1** A interface deve ser intuitiva e acessível.
* **3.2.3.2** Cadastro/login devem ser concluídos em até 30 segundos.
* **3.2.3.3** A busca deve permitir filtros por localização, porte do animal, preço e avaliação.
* **3.2.3.4** O sistema deve oferecer acessibilidade (alto contraste, leitura de tela, fontes ajustáveis).

### 3.2.4 Requisitos de Confiabilidade

* **3.2.4.1** O sistema deve garantir disponibilidade de 99,9%.
* **3.2.4.2** Backups devem ser realizados diariamente com retenção de 12 meses.
* **3.2.4.3** O sistema deve ter recuperação em até 15 minutos após falha crítica.
* **3.2.4.4** O sistema deve notificar a equipe técnica em caso de falha grave.

### 3.2.5 Requisitos de Portabilidade

* **3.2.5.1** O aplicativo deve ser compatível com Android (8.0+) e iOS (12+).
* **3.2.5.2** O app deve ter no máximo 100MB e baixo consumo de bateria.
* **3.2.5.3** O sistema deve ser desenvolvido em framework multiplataforma (React Native ou Flutter).

### 3.2.6 Requisitos de Eficiência

* **3.2.6.1** O sistema deve responder consultas em até 2 segundos.
* **3.2.6.2** O sistema deve suportar até 100 mil usuários simultâneos.
* **3.2.6.3** O desempenho deve ser monitorado continuamente.

### 3.2.7 Requisitos de Espaço

* **3.2.7.1** O sistema deve suportar crescimento de base de dados (corridas, pets, motoristas).
* **3.2.7.2** O sistema deve ser escalável horizontalmente (servidores adicionais sob demanda).
* **3.2.7.3** O sistema deve ser compatível com armazenamento em nuvem híbrida.

1. **Projeto**
   1. **Arquitetura Lógica**

# ARQUITETURA DE SOFTWARE

Este capítulo especifica a arquitetura do sistema de transporte especializado para animais de estimação, detalhando sua composição lógica e física, além das justificativas para a escolha dos componentes, garantindo escalabilidade, manutenção facilitada e atendimento às necessidades do usuário.

## 5.1 ARQUITETURA DE SOFTWARE (LÓGICA)

O sistema será desenvolvido com **arquitetura baseada em microserviços**, permitindo que cada componente funcione de forma independente, mas integrado, garantindo escalabilidade, modularidade e facilidade de manutenção. A comunicação entre front-end e back-end será feita por **APIs RESTful**, assegurando interoperabilidade entre aplicativos móveis, portal administrativo e serviços de terceiros (mapas, pagamento, notificações).

A interface será desenvolvida com foco em **usabilidade e acessibilidade**, seguindo padrões de design responsivo, para proporcionar experiência intuitiva a todos os usuários, incluindo donos de pets e motoristas.

### 5.1.1 Layers do Sistema

As camadas do sistema foram organizadas para distribuir funcionalidades específicas de acordo com o tipo de usuário, garantindo clareza lógica e eficiência na implementação:

* **Login do usuário:** cadastro, login via e-mail, senha, Google e autenticação multifator, edição de perfil e exclusão de conta.
* **Cadastro e perfil do pet:** inclusão de informações detalhadas sobre o animal (raça, porte, idade, vacinas, necessidades especiais).
* **Solicitação e agendamento de corridas:** interface para transporte imediato ou agendado, escolha de veículos adaptados e cálculo de rotas em tempo real.
* **Geolocalização e roteirização:** integração com APIs de mapas para rastreamento de motoristas e otimização de rotas.
* **Sistema de pagamento:** pagamentos via Pix, cartão de crédito/débito ou carteira digital, com segurança e transparência.
* **Avaliações e feedback:** registro de avaliações de motoristas e usuários para garantir qualidade do serviço.
* **Histórico de serviços:** acompanhamento do status de corridas, cancelamentos e histórico de pagamentos.
* **Painel administrativo:** gestão de motoristas, corridas, faturamento, suporte e relatórios estratégicos.
* **Notificações e comunicação:** envio de alertas via push, SMS, e-mail e chat criptografado para suporte e comunicação direta.
* **Segurança e conformidade:** protocolos técnicos para garantir bem-estar dos pets, proteção de dados (LGPD) e conformidade com normas de segurança (ISO/IEC 27001).

### 5.1.2 Justificativa da Arquitetura

A escolha da arquitetura baseada em microserviços permite:

* Escalabilidade horizontal, com adição de serviços conforme crescimento do sistema;
* Independência entre módulos, facilitando atualizações e manutenção sem impactar toda a aplicação;
* Implementação ágil, alinhada à metodologia Scrum, com entregas incrementais e feedback contínuo;
* Integração com APIs externas (pagamento, mapas, notificações) sem comprometer a estabilidade do sistema;
* Modularidade, permitindo futuras implementações de IoT ou novos serviços relacionados ao transporte pet.

### 5.1.2 Tier

O **Tier** é o modelo de arquitetura que realiza a separação do aplicativo em partes, combinando conceitos físicos e lógicos. Essa divisão permite que diferentes equipes trabalhem de forma independente em cada camada, garantindo que alterações em uma camada não causem impacto nas demais.

Para este projeto, selecionamos o modelo **N-Tier**, derivado do tradicional **3-Tier**. No N-Tier, além da divisão básica em três camadas, é possível adicionar outras camadas para melhorar desempenho, escalabilidade e organização do sistema. A divisão do projeto foi organizada da seguinte forma para ambos os aplicativos (usuário e motorista):

* **Apresentação:** camada visível ao usuário, responsável por coletar informações e interagir diretamente com o aplicativo.
* **Aplicativo:** camada responsável por processar as informações recebidas da camada de apresentação, realizando lógica de negócio e regras de transporte.
* **Banco de dados:** camada onde os dados são armazenados em um servidor NoSQL (Firebase), garantindo persistência e segurança das informações.
* **Cache:** banco de memória temporário usado para armazenar informações acessadas com frequência, acelerando o acesso sem necessidade de consultar o banco de dados principal. Deve ser utilizado para itens que não requerem atualizações constantes, como avaliações, portfólios profissionais, lista de serviços, mensagens automáticas e FAQ.
* **API Gateway:** camada que gerencia a comunicação entre o aplicativo principal e outras plataformas, controlando o tráfego de informações externas.
  1. **Arquitetura Física**

Neste item deve ser apresentada a arquitetura de infraestrutura do sistema, demonstrando o tipo de arquitetura física, a configuração de hardware, de rede... Para a representação da arquitetura de infraestrutura pode-se utilizar o diagrama de implantação.

Referência: UC Modelos, métodos e técnicas da engenharia de software

1. **Protótipo de Interface**

Neste item deve ser apresentado o protótipo do projeto. O protótipo é um recurso que deve ser adotado como estratégia para levantamento, detalhamento, validação de requisitos e modelagem de interface com o usuário (usabilidade).

Referência: UC Modelos, métodos e técnicas da engenharia de software

1. **Critérios de Qualidade de Software**

Neste item devem ser listados e descritos os critérios de garantia da qualidade do processo que serão considerados no Projeto.

Referência: UC Gestão e Qualidade de software

**7. Testes**

**7.1. Plano de Testes**

Neste item deve ser criado o plano de testes do sistema, permitindo a validação do sistema por parte do desenvolvedor, através da verificação dos requisitos do sistema desenvolvido.

Referência: UC Gestão e Qualidade de software

**7.2. Roteiro de Testes**

Neste item devem ser registrados os testes realizados no sistema tendo como base o Plano de Testes do Sistema. O roteiro de testes deve ser elaborado com base nos casos de uso ou user stories (cartão de histórias) de forma manual ou automatizada.

Referência: UC Gestão e Qualidade de software

**Anexo I**

Neste item deve ser anexado o roteiro de entrevista ou questionário respondido, caso tenha sido aplicado.

Referência: UC Modelos, métodos e técnicas da engenharia de software