# DOCUMENTO DE ESPECIFICAÇÃO E MODELAGEM DO SISTEMA SHARECONDO

**Projeto:** ShareCondo

**Integrantes:** Caio Rocha, Guilherme Spinola

**Disciplina:** Linguagem de Programação II & Modelagem de Software

**Semestre:** 6º

**Data:** 01 de junho de 2025

## Sumário

1. Introdução

1.1. Propósito do Documento

1.2. Escopo do Projeto

1.3. Definições, Acrônimos e Abreviações

1.4. Referências

1.5. Visão Geral do Documento

1. Descrição Geral

2.1. Perspectiva do Produto

2.2. Funções do Produto

2.3. Características dos Usuários

2.4. Restrições

2.5. Suposições e Dependências

1. Requisitos Específicos

3.1. Requisitos Funcionais

3.2. Requisitos Não-Funcionais

3.3. Requisitos de Interface

3.4. Requisitos de Dados

1. Modelagem de Processos de Negócio (BPMN)
2. Modelagem UML e Diagramas

5.1. Modelo de Domínio

5.2. Diagrama de Casos de Uso

5.3. Diagrama de Sequência

5.4. Diagrama de Classes Detalhado

1. Protótipos e Wireframes

6.1. Wireframes

6.2. Storyboard

6.3. Story Map

1. Análise de Prioridade de Recursos (Modelo Kano)
2. Casos de Teste Funcionais
3. Matrizes de Rastreabilidade

9.1. Matriz de Rastreabilidade: Requisitos vs. Objetivos

9.2. Matriz de Rastreabilidade: Requisitos vs. Casos de Teste

1. Referências Bibliográficas e Ferramentas

## 1. INTRODUÇÃO

### 1.1 Propósito do Documento

Este documento tem como objetivo definir e especificar os requisitos funcionais, não-funcionais, de interface e de dados do sistema **ShareCondo**. Ele servirá como base para o desenvolvimento, testes e validação do produto, estabelecendo um acordo entre as partes interessadas sobre o que o sistema deverá fazer.

### 1.2 Escopo do Projeto

O ShareCondo é uma plataforma projetada para conectar vizinhos dentro de um condomínio, permitindo que eles ofereçam ou solicitem diversas habilidades, serviços e itens. O objetivo principal é fomentar um ambiente colaborativo, onde os moradores possam realizar trocas, empréstimos ou contratar pequenos serviços entre si, como reparos, jardinagem, aulas particulares, ou prática de idiomas.

**Dentro do Escopo:**

Gerenciamento de Usuários (Cadastro, Login, Aprovação de Moradores).

Gerenciamento de Condomínios (Criação e listagem).

Gerenciamento de Anúncios (Criação, listagem, visualização dos próprios anúncios).

Gerenciamento de Ofertas (Criação de ofertas para anúncios, aceitação/recusa de ofertas).

Sistema de autenticação e autorização baseado em papéis (ADMIN, SINDICO, USUARIO).

Interface front-end para interação do usuário e administração.

API back-end para processamento das lógicas de negócio e persistência de dados.

**Fora do Escopo:**

Sistema de pagamento integrado para transações monetárias (as negociações financeiras são de responsabilidade dos usuários).

Sistema de avaliação e reputação de usuários.

Comunicação em tempo real (chat) entre usuários (interações são baseadas em ofertas e dados de contato).

Aplicativo móvel nativo (o foco é uma aplicação web responsiva).

### 1.3 Definições, Acrônimos e Abreviações

| Termo | Definição |
| --- | --- |
| RF | Requisito Funcional |
| RNF | Requisito Não-Funcional |
| RI | Requisito de Interface |
| RD | Requisito de Dados |
| API | Application Programming Interface (Interface de Programação de Aplicações) |
| JWT | JSON Web Token (Token Web JSON para autenticação) |
| CRUD | Create, Read, Update, Delete (Operações básicas de dados) |
| DTO | Data Transfer Object (Objeto de Transferência de Dados) |
| UI | User Interface (Interface do Usuário) |
| UX | User Experience (Experiência do Usuário) |
| BPMN | Business Process Model and Notation (Modelo e Notação de Processos de Negócio) |
| UML | Unified Modeling Language (Linguagem de Modelagem Unificada) |
| SQL | Structured Query Language (Linguagem de Consulta Estruturada) |
| HTTP | Hypertext Transfer Protocol |
| HTTPS | Hypertext Transfer Protocol Secure |
| JSON | JavaScript Object Notation |

### 1.4 Referências

* Documentação do Spring Boot: https://spring.io/projects/spring-boot
* Documentação do React: https://react.dev/
* Documentação do JWT (java-jwt): https://github.com/auth0/java-jwt
* Padrões de API RESTful.
* Arquivos do projeto ShareCondo (back-end e front-end).
* MODELO DE DOCUMENTAÇÃO DE REQUISITOS.docx

### 1.5 Visão Geral do Documento

Este documento está organizado da seguinte forma:

**Seção 1: Introdução** – Fornece uma visão geral do documento.

**Seção 2: Descrição Geral** – Apresenta uma visão de alto nível do produto.

**Seção 3: Requisitos Específicos** – Detalha os requisitos funcionais e não-funcionais.

**Seção 4: Modelagem de Processos de Negócio (BPMN)** – Descreve processos críticos do sistema.

**Seção 5: Modelagem UML e Diagramas** – Inclui diagramas UML para modelagem do sistema.

**Seção 6: Protótipos e Wireframes** – Apresenta representações visuais da interface.

**Seção 7: Análise de Prioridade de Recursos** – Analisa funcionalidades chave.

**Seção 8: Casos de Teste Funcionais** – Define testes para as funcionalidades.

**Seção 9: Matrizes de Rastreabilidade** – Mapeia requisitos a objetivos e casos de teste.

**Seção 10: Glossário** – Lista termos e abreviações.

**Seção 11: Referências Bibliográficas e Ferramentas** – Cita fontes e ferramentas.

## 2. DESCRIÇÃO GERAL

### 2.1 Perspectiva do Produto

O ShareCondo é um sistema independente, projetado para ser uma plataforma web completa. Ele possui uma arquitetura cliente-servidor, com um front-end desenvolvido em React e um back-end em Java com Spring Boot. A comunicação entre o cliente e o servidor ocorre via API RESTful. O sistema utiliza um banco de dados SQL Server para persistência de dados.

### 2.2 Funções do Produto

As principais funções do sistema ShareCondo incluem:

**Autenticação e Gerenciamento de Perfis:**

* Permitir o cadastro de novos usuários (Moradores, Síndicos, Administradores).
* Autenticar usuários através de login (email e senha).
* Permitir que usuários atualizem seus próprios dados cadastrais.

**Gerenciamento de Condomínios:**

* Permitir que Administradores cadastrem novos condomínios.
* Listar condomínios para seleção durante o cadastro de Moradores e Síndicos.

**Aprovação de Usuários:**

* Permitir que Síndicos aprovem ou rejeitem cadastros de Moradores para seus respectivos condomínios.
* Permitir que Administradores aprovem ou rejeitem quaisquer usuários pendentes.

**Criação e Gerenciamento de Anúncios:**

* Permitir que usuários (aprovados) criem anúncios de itens ou serviços.
* Permitir que usuários visualizem anúncios (filtrados por condomínio, exceto para Admin).
* Permitir que usuários visualizem e gerenciem seus próprios anúncios.

**Criação e Gerenciamento de Ofertas:**

* Permitir que usuários (aprovados) façam ofertas em anúncios de outros usuários (dentro do mesmo condomínio, exceto Admin).
* Permitir que o anunciante aceite ou recuse ofertas recebidas.
* Visualizar ofertas feitas e recebidas.

**Administração do Sistema (Painel Admin):**

* Gerenciar usuários (listar, editar, excluir).
* Gerenciar condomínios (listar, editar, excluir).
* Gerenciar anúncios (listar, editar, excluir).
* Gerenciar ofertas (visualizar).

### 2.3 Características dos Usuários

O sistema será utilizado pelos seguintes tipos de usuários:

* **Morador (USUARIO):**
  + **Nível Educacional:** Variado.
  + **Experiência:** Usuário comum de internet e aplicações web.
  + **Conhecimento Técnico:** Baixo a médio.
  + **Funções Principais:** Cadastrar-se, visualizar anúncios do seu condomínio, criar anúncios, fazer/receber ofertas.
* **Síndico (SINDICO):**
  + **Nível Educacional:** Variado, geralmente com alguma experiência em gestão.
  + **Experiência:** Usuário de internet e aplicações web, possivelmente com alguma experiência em sistemas de gerenciamento.
  + **Conhecimento Técnico:** Médio.
  + **Funções Principais:** Todas as funções de Morador, mais aprovar/rejeitar cadastros de moradores do seu condomínio.
* **Administrador (ADMIN):**
  + **Nível Educacional:** Geralmente com formação técnica ou superior.
  + **Experiência:** Avançada em sistemas web e gerenciamento de plataformas.
  + **Conhecimento Técnico:** Alto.
  + **Funções Principais:** Gerenciamento completo de usuários, condomínios, anúncios e ofertas de toda a plataforma.

### 2.4 Restrições

* A autenticação é obrigatória para a maioria das funcionalidades.
* Usuários do tipo “USUARIO” só podem criar um número limitado de anúncios ativos (MAX\_ACTIVE\_ANNOUNCEMENTS\_PER\_USER = 5).
* A criação de ofertas entre usuários (não ADMIN) é restrita a membros do mesmo condomínio.
* A senha dos usuários deve ser armazenada de forma criptografada (BCrypt).
* O acesso às funcionalidades da API deve ser protegido por JWT e verificação de papéis.

### 2.5 Suposições e Dependências

* **Suposições:**
  + Os usuários possuem acesso à internet e um navegador web.
  + Os usuários fornecerão informações verídicas durante o cadastro.
  + Os síndicos realizarão a aprovação de moradores de forma diligente.
  + A infraestrutura do servidor (onde o back-end e o banco de dados estão hospedados) é estável e disponível.
* **Dependências:**
  + Funcionamento do servidor de banco de dados (SQL Server).
  + Disponibilidade do servidor de aplicação Java (Spring Boot).
  + Bibliotecas de terceiros (listadas no pom.xml e package.json) devem funcionar conforme esperado.

**3. REQUISITOS ESPECÍFICOS**

**3.1 Requisitos Funcionais**

**RF-001: Cadastro de Usuário**

* **Descrição:** O sistema deve permitir que um novo usuário se cadastre fornecendo nome, email, senha e, se for do tipo MORADOR ou SINDICO, selecionando um condomínio existente.
* **Critérios de Aceitação:** Sucesso no cadastro com dados válidos, email único, senha confirmada, condomínio obrigatório para MORADOR/SINDICO. MORADOR inicia como "PENDENTE\_APROVACAO"; ADMIN e SÍNDICO são "APROVADO" automaticamente.
* **Dependências:** N/A.

**RF-002: Login de Usuário**

* **Descrição:** O sistema deve permitir que um usuário registrado acesse o sistema fornecendo email e senha.
* **Critérios de Aceitação:** Login bem-sucedido com credenciais válidas, geração de token JWT, armazenamento de dados no cliente (localStorage). Usuários MORADOR com status diferente de "APROVADO" são impedidos de logar.
* **Dependências:** RF-001.

**RF-003: Logout de Usuário**

* **Descrição:** O sistema deve permitir que um usuário logado encerre sua sessão.
* **Critérios de Aceitação:** Opção de "Sair" disponível, remoção de token/dados do armazenamento local, redirecionamento para página inicial.
* **Dependências:** RF-002.

**RF-004: Criação de Condomínio (Admin)**

* **Descrição:** O sistema deve permitir que um usuário ADMIN crie um novo condomínio informando nome e endereço.
* **Critérios de Aceitação:** Um ADMIN logado deve conseguir submeter o formulário de criação de condomínio; o novo condomínio deve ser persistido no banco de dados.
* **Dependências:** RF-002 (Login de Admin).

**RF-005: Aprovação/Rejeição de Cadastro de Morador (Síndico/Admin)**

* **Descrição:** O sistema deve permitir que um SINDICO aprove ou rejeite o cadastro de moradores pendentes em seu condomínio. ADMINs podem aprovar/rejeitar qualquer usuário pendente.
* **Critérios de Aceitação:** SINDICO logado deve visualizar uma lista de usuários pendentes de seu condomínio; ADMIN logado deve visualizar uma lista de todos os usuários pendentes. SINDICO/ADMIN deve conseguir aprovar um cadastro pendente, mudando seu status para "APROVADO"; SINDICO/ADMIN deve conseguir rejeitar um cadastro pendente, mudando seu status para "REJEITADO".
* **Dependências:** RF-001, RF-002.

**RF-006: Criação de Anúncio**

* **Descrição:** O sistema deve permitir que um usuário logado e APROVADO crie um anúncio de item ou serviço, fornecendo título, descrição e tipo.
* **Critérios de Aceitação:** Usuário logado e APROVADO deve conseguir submeter o formulário de criação de anúncio; o anúncio deve ser associado ao usuário logado. Usuários do tipo USUARIO têm um limite de 5 anúncios ativos. O anúncio é criado como "ATIVO" por padrão.
* **Dependências:** RF-002, (RF-005 ou auto-aprovação para Síndico/Admin).

**RF-007: Listagem de Anúncios**

* **Descrição:** O sistema deve listar anúncios ativos. Usuários comuns e síndicos veem apenas anúncios de seu condomínio. ADMINs veem todos.
* **Critérios de Aceitação:** Usuários logados e APROVADOS podem ver a lista de anúncios. ADMINs veem todos os anúncios ativos. SINDICO e USUARIO veem apenas anúncios ativos de usuários de seu próprio condomínio.
* **Dependências:** RF-006.

**RF-008: Visualização de Detalhes do Anúncio**

* **Descrição:** O sistema deve permitir que um usuário logado e APROVADO visualize os detalhes completos de um anúncio.
* **Critérios de Aceitação:** Usuários logados e APROVADOS podem acessar a página de detalhes de um anúncio; a visualização é restrita ao condomínio do usuário, exceto para ADMIN. Todos os dados relevantes do anúncio (título, descrição, tipo, anunciante, data, status) são exibidos.
* **Dependências:** RF-007.

**RF-009: Gerenciamento dos Próprios Anúncios**

* **Descrição:** O sistema deve permitir que um usuário logado e APROVADO visualize, edite e delete seus próprios anúncios.
* **Critérios de Aceitação:** Usuário pode listar todos os seus anúncios (ativos e inativos). Usuário pode editar título, descrição, tipo e status (ativo/inativo) de seus anúncios. Usuário pode excluir seus anúncios.
* **Dependências:** RF-006.

**RF-010: Criação de Oferta**

* **Descrição:** O sistema deve permitir que um usuário logado e APROVADO faça uma oferta em um anúncio ativo de outro usuário (do mesmo condomínio, exceto se um dos envolvidos for ADMIN).
* **Critérios de Aceitação:** Usuário logado e APROVADO pode submeter uma oferta; não é possível ofertar no próprio anúncio. Ofertas entre USUARIOs/SINDICO são restritas ao mesmo condomínio (a menos que um seja ADMIN). A oferta pode ser do tipo DINHEIRO (com valor) ou ITEM/SERVICO (com descrição). A oferta é criada com status "PENDENTE".
* **Dependências:** RF-007.

**RF-011: Gerenciamento de Ofertas (Aceitar/Recusar)**

* **Descrição:** O sistema deve permitir que o anunciante aceite ou recuse uma oferta PENDENTE para seu anúncio ativo.
* **Critérios de Aceitação:**
  + **Aceitar:** Apenas o anunciante pode aceitar; oferta deve ser "PENDENTE"; anúncio deve estar "ATIVO". Ao aceitar, o status da oferta muda para "ACEITA"; o anúncio relacionado é desativado (status "INATIVO"); outras ofertas PENDENTES para o mesmo anúncio são automaticamente RECUSADAS.
  + **Recusar:** Apenas o anunciante pode recusar; a oferta deve estar com status "PENDENTE". Ao recusar, o status da oferta muda para "RECUSADA".
* **Dependências:** RF-010.

**RF-012: Visualização de Minhas Ofertas (Feitas)**

* **Descrição:** O sistema deve permitir que um usuário logado e APROVADO visualize as ofertas que ele fez em anúncios de outros.
* **Critérios de Aceitação:** Usuário logado e APROVADO pode ver a lista de suas ofertas enviadas. A lista deve exibir detalhes da oferta (anúncio alvo, tipo, valor/descrição, data, status).
* **Dependências:** RF-010.

**RF-013: Gerenciamento de Entidades (Admin)**

* **Descrição:** Usuários ADMIN devem ter acesso a funcionalidades CRUD completas para Usuários, Condomínios e Anúncios. Para Ofertas, a funcionalidade é primariamente de visualização.
* **Critérios de Aceitação:** ADMIN pode criar, listar, visualizar por ID, atualizar e excluir qualquer Usuário, Condomínio e Anúncio. ADMIN pode listar e visualizar ofertas (filtrando por anúncio ou usuário).
* **Dependências:** RF-002.

**3.2 Requisitos Não-Funcionais**

**RNF-001: Usabilidade - Interface Intuitiva**

* **Descrição:** A interface do usuário deve ser clara, intuitiva e fácil de usar para todos os tipos de usuários, minimizando a curva de aprendizado.
* **Critérios de Aceitação:** Novos usuários conseguem se cadastrar e criar um anúncio em menos de 5 minutos sem necessidade de manual. As principais funcionalidades devem ser acessíveis com no máximo 3 cliques a partir da página principal (após login). Feedback visual claro (mensagens de sucesso, erro, indicadores de carregamento) deve ser fornecido para todas as ações do usuário.

**RNF-002: Segurança - Proteção de Dados do Usuário**

* **Descrição:** Os dados dos usuários, especialmente senhas e informações pessoais, devem ser protegidos contra acesso não autorizado.
* **Critérios de Aceitação:** Senhas devem ser armazenadas usando hashing BCrypt. A comunicação entre cliente e servidor deve utilizar HTTPS em ambiente de produção. Acesso às funcionalidades da API deve ser protegido por autenticação (JWT) e autorização baseada em papéis (ROLE\_ADMIN, ROLE\_SINDICO, ROLE\_USER). Tokens JWT devem ter tempo de expiração configurado (ex: 1 hora, conforme TokenService.java).

**RNF-003: Segurança - Validação de Entrada**

* **Descrição:** Todas as entradas de dados, tanto no front-end quanto no back-end, devem ser validadas para prevenir ataques (ex: XSS, SQL Injection) e garantir a integridade dos dados.
* **Critérios de Aceitação:** Dados de entrada nas APIs são validados quanto ao tipo, formato, tamanho e obrigatoriedade (conforme DTOs e lógica nos controllers). Entradas de texto que serão exibidas na UI são tratadas para prevenir Cross-Site Scripting (XSS) (responsabilidade do front-end ao renderizar). Consultas ao banco de dados utilizam Prepared Statements ou ORM (JPA) que previne SQL Injection.

**RNF-004: Manutenibilidade**

* **Descrição:** O código do sistema deve ser bem estruturado, comentado e seguir boas práticas de desenvolvimento para facilitar a manutenção e futuras evoluções.
* **Critérios de Aceitação:** O código segue os padrões de nomenclatura e estilo definidos pela equipe; componentes e módulos são razoavelmente desacoplados; cobertura de testes unitários no back-end de pelo menos 60%; comentários explicativos em trechos complexos do código.

**RNF-005: Escalabilidade**

* **Descrição:** O sistema deve ser capaz de lidar com um aumento no número de usuários, condomínios, anúncios e ofertas sem degradação significativa de desempenho.
* **Critérios de Aceitação:** A arquitetura do back-end (Spring Boot) permite escalonamento horizontal. O banco de dados (SQL Server) suporta otimizações e indexações para lidar com aumento de volume de dados.

**3.3 Requisitos de Interface**

**RI-001: Interface Web do Usuário (Front-end React)**

* **Descrição:** O sistema deve prover uma interface gráfica web responsiva para interação dos usuários (Moradores, Síndicos) e Administradores.
* **Tipo de Interface:** Usuário
* **Prioridade:** Alta
* **Fonte:** Arquitetura do projeto (React).
* **Especificações:**
  1. **Página Inicial (HomePage.js):** Apresentação do ShareCondo, seções "Como Funciona", "Benefícios" e CTAs. Layout conforme HomePage.css.
  2. **Página de Cadastro (CadastroPage.js):** Formulário para registro de novos usuários, incluindo seleção de condomínio. Layout conforme Formulario.css.
  3. **Página de Login (LoginPage.js):** Formulário para autenticação de usuários. Layout conforme Formulario.css.
  4. **Página de Anúncios (AnunciosPage.js):** Listagem de anúncios, filtros (implícito), botão para criar novo anúncio. Layout conforme Anuncio.css.
  5. **Página de Detalhes do Anúncio (AnuncioDetalhePage.js):** Exibição completa dos dados de um anúncio, opção de fazer oferta. Layout conforme AnuncioDetalhe.css.
  6. **Página "Meus Anúncios" (MeusAnunciosPage.js):** Listagem dos anúncios criados pelo usuário logado, com opções de gerenciamento de ofertas recebidas. Layout conforme MeusAnuncios.css.
  7. **Página "Minhas Ofertas" (MinhasOfertasPage.js):** Listagem das ofertas feitas pelo usuário logado e seus status. Layout conforme MinhasOfertas.css.
  8. **Página "Aprovar Usuários" (Síndico) (AprovarUsuariosPage.js):** Interface para síndicos visualizarem e gerenciarem usuários pendentes de aprovação em seu condomínio. Layout conforme AprovarUsuarios.css.
  9. **Painel Administrativo (AdminLayout.js e subpáginas):** Interface para administradores gerenciarem usuários, condomínios, anúncios e ofertas. Layouts conforme AdminLayout.css, CrudTabela.css, AdminModal.css.
  10. **Componentes Reutilizáveis:** Cabeçalho (Cabecalho/index.js, Cabecalho.css), Rodapé (Rodape/Rodape.js, Rodape.css), Botões (Botao/Button.js, Button.css), Modais (Modal/Modal.js, Modal.css; Anuncios/AddAnuncioModal.js; Anuncios/OfferTradeModal.js; Admin/AddCondoModal.js; Admin/EditUserModal.js), Formulários (Formulario/index.js, Formulario.css; FormInput/index.js, FormInput.css), Tabelas CRUD (CrudTabela/index.js, CrudTabela.css).
  11. **Página "Quem Somos" (QuemSomosPage.js, QuemSomosPage.css):** Página informativa sobre o projeto.
  12. **Página "Não Encontrado" (NotFoundPage.js):** Página para rotas inválidas.

**RI-002: API RESTful do Back-end (Spring Boot)**

* **Descrição:** O sistema deve expor uma API RESTful para comunicação entre o front-end e o back-end, gerenciando todas as operações de negócio.
* **Tipo de Interface:** Software (API)
* **Prioridade:** Alta
* **Fonte:** Arquitetura do projeto (Spring Boot).
* **Especificações:**
  1. **Endpoints de Autenticação (/auth):** POST /login (autentica usuário e retorna JWT), POST /register (registra novo usuário).
  2. **Endpoints de Usuário (/usuario):** GET / (lista todos os usuários, Admin), GET /{id} (busca usuário por ID, Autenticado), PUT /{id} (atualiza usuário, Admin ou proprietário), PATCH /{id} (atualiza parcialmente usuário, Admin ou proprietário), DELETE /{id} (deleta usuário, Admin), GET /pendentes (lista usuários pendentes de aprovação, Síndico/Admin), PATCH /{id}/aprovar (aprova usuário, Síndico/Admin), PATCH /{id}/rejeitar (rejeita usuário, Síndico/Admin).
  3. **Endpoints de Condomínio (/condominio):** POST / (cria condomínio, Admin), GET / (lista todos os condomínios, PermitAll), GET /{id} (busca condomínio por ID, PermitAll), PUT /{id} (atualiza condomínio, Admin), DELETE /{id} (deleta condomínio, Admin).
  4. **Endpoints de Anúncio (/anuncios):** POST / (cria anúncio, Usuário aprovado), GET / (lista anúncios, filtrado por condomínio ou todos para Admin), GET /meus (lista anúncios do usuário logado), GET /{id} (busca anúncio por ID, restrições de condomínio aplicáveis), PUT /{id} (atualiza anúncio, Anunciante ou Admin), DELETE /{id} (deleta anúncio, Anunciante ou Admin).
  5. **Endpoints de Oferta (/ofertas):** POST /anuncio/{anuncioId} (cria oferta para um anúncio, Usuário aprovado), GET /anuncio/{anuncioId} (lista ofertas de um anúncio), GET /usuario/{usuarioId} (lista ofertas feitas por um usuário), PATCH /{ofertaId}/aceitar (Anunciante aceita oferta), PATCH /{ofertaId}/recusar (Anunciante recusa oferta), GET / (lista todas as ofertas, Admin/Síndico com restrições).
  6. **Formato de Dados:** JSON.
  7. **Autenticação:** JWT Bearer Token no header Authorization.
  8. **CORS:** Configurado em CorsConfig.java para permitir requisições de http://localhost:3000 com métodos GET, POST, PUT, DELETE, PATCH e todos os headers.
* **Dependências:** RNF-002, RNF-003.

**3.4 Requisitos de Dados**

**RD-001: Dados do Usuário**

* **Descrição:** Informações sobre os usuários cadastrados no sistema.
* **Prioridade:** Alta
* **Fonte:** Usuario.java, UsuarioDTO.java, RegisterDTO.java.
* **Especificações:**
  + **Tipo de dados:** ID (String UUID), Nome (String), Email (String, único), Senha (String, criptografada), TipoUsuario (Enum: ADMIN, SINDICO, USUARIO), StatusUsuario (Enum: PENDENTE\_APROVACAO, APROVADO, REJEITADO), Condominio (FK para Condominio).
  + **Volume:** Estimado em 1000-10000 usuários inicialmente, com crescimento.
  + **Frequência de acesso:** Alta (login, visualização de perfis, associação a anúncios/ofertas).
  + **Requisitos de retenção:** Indefinido (manter enquanto a conta existir, sujeito a políticas de privacidade).
  + **Requisitos de backup:** Backup diário do banco de dados.
* **Dependências:** RD-002 (Condominio).

**RD-002: Dados do Condomínio**

* **Descrição:** Informações sobre os condomínios cadastrados.
* **Prioridade:** Alta
* **Fonte:** Condominio.java, CondominioDTO.java.
* **Especificações:**
  + **Tipo de dados:** ID (String UUID), Nome (String), Endereco (String).
  + **Volume:** Estimado em 10-100 condomínios inicialmente, com crescimento moderado.
  + **Frequência de acesso:** Média (cadastro de usuários, listagem, filtros de anúncios).
  + **Requisitos de retenção:** Indefinido (manter enquanto o condomínio existir na plataforma).
  + **Requisitos de backup:** Backup diário do banco de dados.
* **Dependências:** N/A.

**RD-003: Dados do Anúncio**

* **Descrição:** Informações sobre os anúncios de itens ou serviços criados pelos usuários.
* **Prioridade:** Alta
* **Fonte:** Anuncio.java, AnuncioRequestDTO.java, AnuncioResponseDTO.java.
* **Especificações:**
  + **Tipo de dados:** ID (String UUID), Titulo (String), Descricao (String), TipoAnuncio (Enum: ITEM, SERVICO), Anunciante (FK para Usuario), DataCriacao (LocalDateTime), Ativo (boolean).
  + **Volume:** Estimado em 5-20 anúncios por usuário ativo.
  + **Frequência de acesso:** Alta (listagem, visualização de detalhes).
  + **Requisitos de retenção:** Manter enquanto o anúncio for relevante ou até ser excluído pelo usuário/admin. Anúncios inativos (após oferta aceita) devem ser mantidos para histórico.
  + **Requisitos de backup:** Backup diário do banco de dados.
* **Dependências:** RD-001 (Usuario).

**RD-004: Dados da Oferta**

* **Descrição:** Informações sobre as ofertas feitas pelos usuários em anúncios.
* **Prioridade:** Alta
* **Fonte:** Oferta.java, OfertaRequestDTO.java, OfertaResponseDTO.java.
* **Especificações:**
  + **Tipo de dados:** ID (String UUID), Anuncio (FK para Anuncio), Ofertante (FK para Usuario), TipoOferta (Enum: DINHEIRO, ITEM, SERVICO), Valor (BigDecimal, opcional), Descricao (String, opcional para item/serviço), DataOferta (LocalDateTime), Status (Enum: PENDENTE, ACEITA, RECUSADA, CANCELADA).
  + **Volume:** Estimado em 0-10 ofertas por anúncio.
  + **Frequência de acesso:** Média a Alta (criação, visualização pelo anunciante e ofertante).
  + **Requisitos de retenção:** Manter enquanto o anúncio e as contas dos usuários envolvidos existirem.
  + **Requisitos de backup:** Backup diário do banco de dados.
* **Dependências:** RD-001 (Usuario), RD-003 (Anuncio).

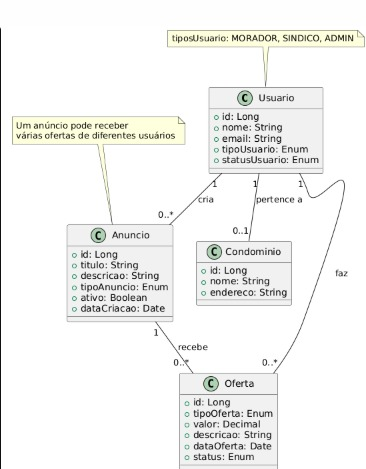
**4. MODELAGEM DE PROCESSOS DE NEGÓCIO (BPMN)**

**Diagrama

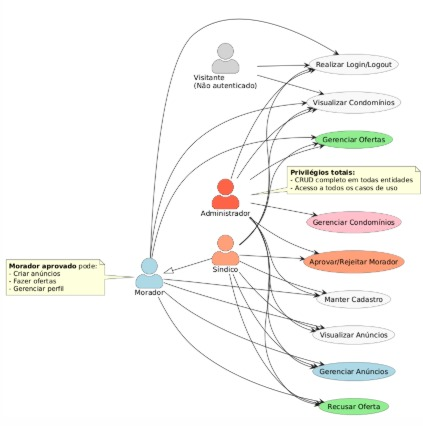
O conteúdo gerado por IA pode estar incorreto.**

**5. MODELAGEM UML E DIAGRAMAS**

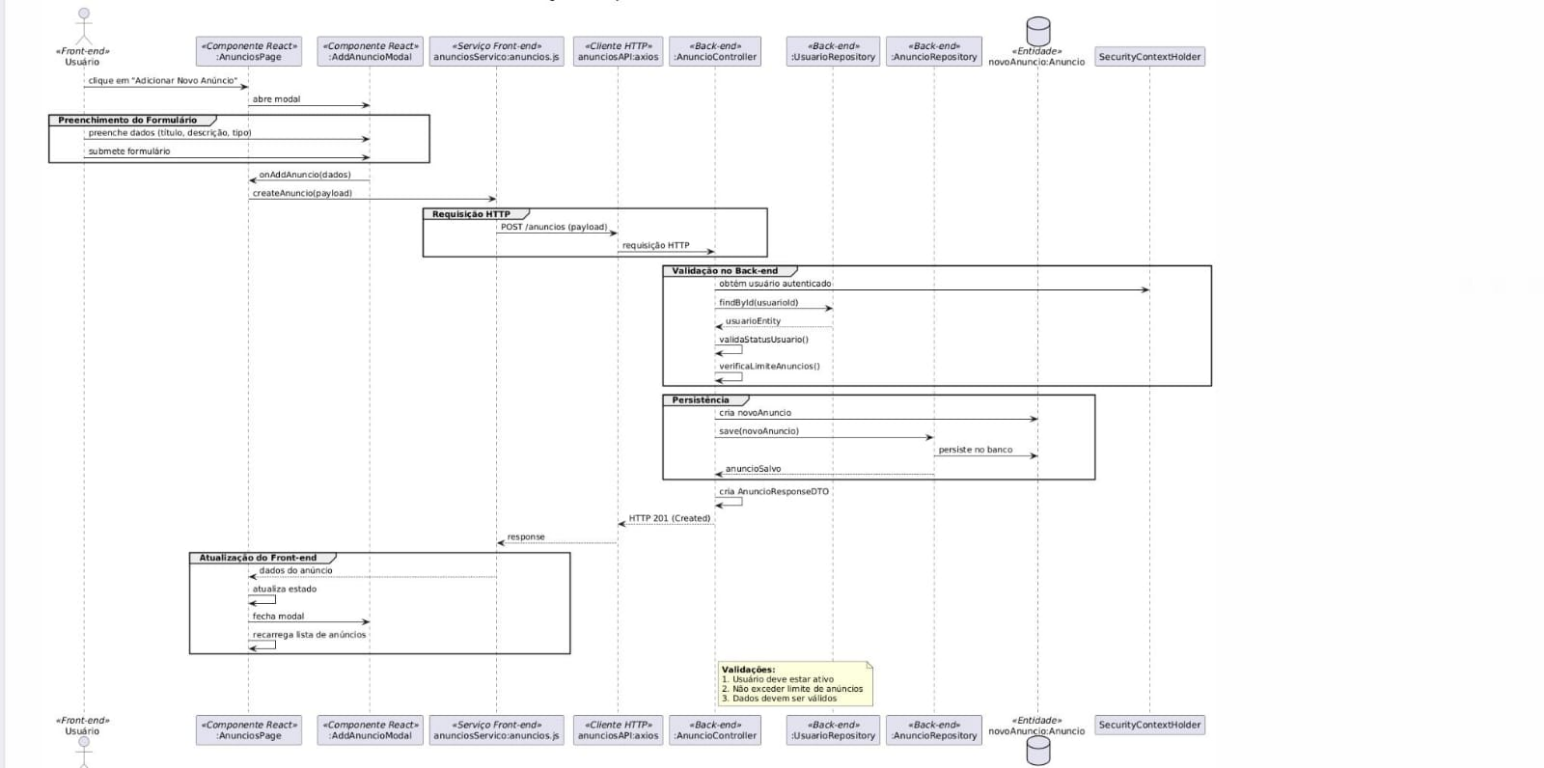
**5.1. Modelo de Domínio (Diagrama de Classes de Alto Nível)**



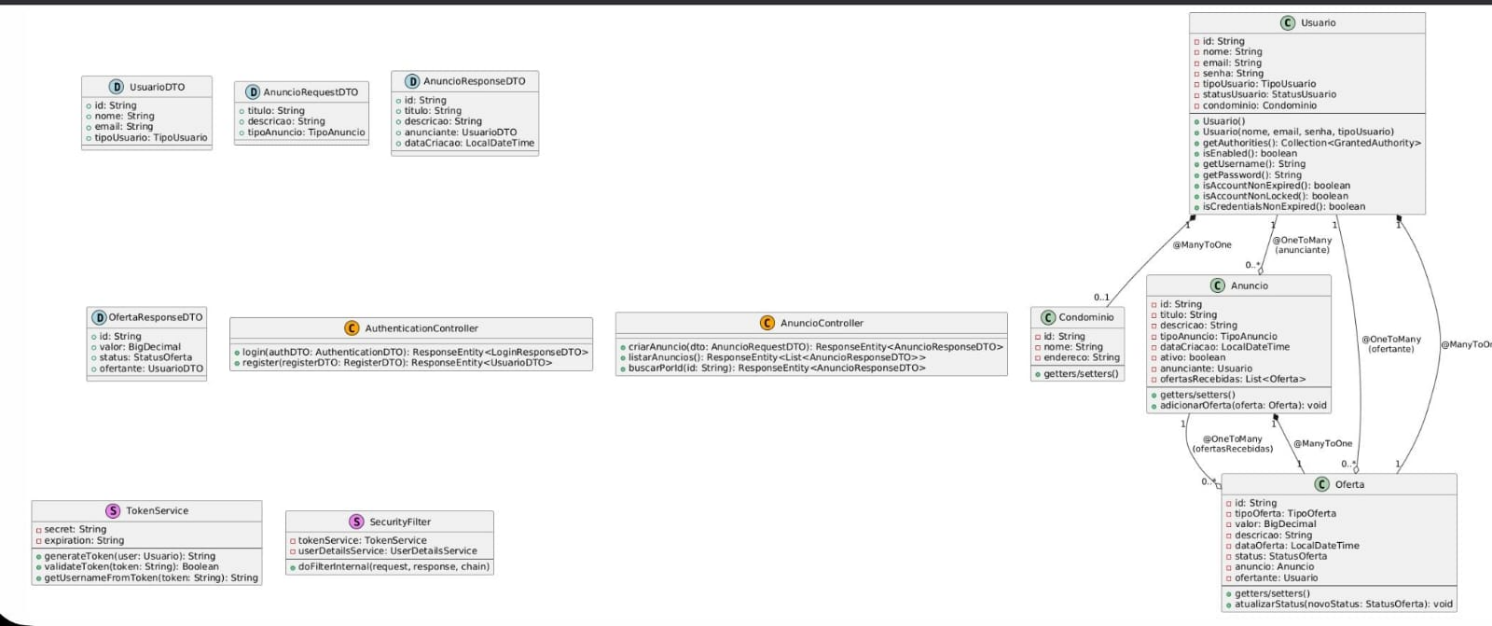
**5.2. Diagrama de Casos de Uso**



**5.3. Diagrama de Sequência**



**5.4. Diagrama de Classes Detalhado**



**6. PROTÓTIPOS E WIREFRAMES**

**6.1. Wireframes** As telas do front-end desenvolvidas em React, localizadas em share-condo-front-end/src/paginas/, servem como protótipos funcionais de média fidelidade para as principais interações do usuário. Cada subdiretório em paginas (e.g., Login, Cadastro, Anuncios, Admin) contém os componentes que formam as telas do sistema.

**Exemplos de Telas Principais (Arquivos de Referência):**

* **Login:** share-condo-front-end/src/paginas/Login/LoginPage.js

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

* **Cadastro:** share-condo-front-end/src/paginas/Cadastro/CadastroPage.js

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

* **Listagem de Anúncios:** share-condo-front-end/src/paginas/Anuncios/AnunciosPage.js

Interface gráfica do usuário, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

* **Painel Admin (Ex: Gerenciamento de Usuários):** share-condo-front-end/src/paginas/Admin/UserManagementPage.js

Interface gráfica do usuário, Texto, Aplicativo

O conteúdo gerado por IA pode estar incorreto.

**6.2. Storyboard: "Usuário Cria Anúncio e Outro Usuário Faz uma Oferta"**

****

**6.3. Story Map**

Interface gráfica do usuário, Texto, Aplicativo, Site

O conteúdo gerado por IA pode estar incorreto.

**7. ANÁLISE DE PRIORIDADE DE RECURSOS (MODELO KANO SIMPLIFICADO)**

Selecionamos três funcionalidades-chave para uma análise simplificada do Modelo Kano:

1. **Funcionalidade-Chave 1: Cadastro e Login de Usuários (RF-001, RF-002)**
   * **Categoria Kano:** **Must-Be (Obrigatória)**
   * **Justificativa:** Sem a capacidade de usuários se registrarem e autenticarem, o sistema não tem como funcionar. É uma expectativa básica e fundamental. A ausência completa dessas funcionalidades causaria extrema insatisfação.
2. **Funcionalidade-Chave 2: Criação e Visualização de Anúncios (RF-006, RF-007)**
   * **Categoria Kano:** **Performance (Unidimensional)**
   * **Justificativa:** A capacidade de criar e ver anúncios é central para o propósito do ShareCondo. Quanto mais fácil, rápido e eficiente for criar um anúncio, e quanto mais relevantes e bem apresentados forem os anúncios listados, maior será a satisfação do usuário. Uma implementação ruim ou limitada causaria insatisfação, enquanto uma boa implementação aumenta a satisfação proporcionalmente.
3. **Funcionalidade-Chave 3: Sistema de Aprovação de Moradores pelo Síndico (RF-005)**
   * **Categoria Kano:** **Attractive (Atratativa) / Performance (para o Síndico)**
   * **Justificativa:**
     + **Para Moradores:** Pode ser vista como **Attractive** ou até mesmo **Must-Be** dependendo da preocupação com segurança no condomínio. Saber que há um controle de quem entra na plataforma específica do condomínio aumenta a confiança e pode ser um diferencial. Se ausente, pode não causar grande insatisfação inicial, mas sua presença agrega valor significativo.
     + **Para Síndicos:** A facilidade e clareza dessa ferramenta é **Performance**. Se for difícil de usar ou confusa, causará insatisfação ao síndico. Se for eficiente, aumenta a satisfação dele com a plataforma.
     + **Para o Negócio/Plataforma:** É um **Must-Be** para garantir a integridade e a proposta de valor de ser uma plataforma "condominial".

**8. CASOS DE TESTE FUNCIONAIS**

**CT-001: Cadastro de Novo Morador com Sucesso**

* **Descrição:** Verificar se um novo usuário do tipo MORADOR consegue se cadastrar com sucesso selecionando um condomínio.
* **Entrada:**
  + Nome: "João Teste Silva"
  + Email: (email único, ex: "joao.teste.novo@exemplo.com")
  + Senha: "senhaValida123"
  + Confirmar Senha: "senhaValida123"
  + Condomínio: Selecionar um condomínio válido da lista (ex: "Residencial Vila das Flores").
* **Procedimento:**
  + Acessar a página de cadastro.
  + Preencher todos os campos do formulário com os dados de entrada.
  + Clicar no botão "Cadastrar".
* **Resultado Esperado:** O sistema exibe uma mensagem de sucesso. O usuário é registrado no banco de dados com o tipo "USUARIO" e status "PENDENTE\_APROVACAO", associado ao condomínio selecionado.

**CT-002: Login de Usuário Aprovado com Sucesso** (Antigo CT-003)

* **Descrição:** Verificar se um usuário com cadastro APROVADO consegue realizar login com sucesso.
* **Entrada:**
  + Email: "sindico.flores@exemplo.com"
  + Senha: "123"
* **Procedimento:**
  + Acessar a página de login.
  + Preencher email e senha.
  + Clicar no botão "Entrar".
* **Resultado Esperado:** Login é bem-sucedido. Usuário é redirecionado para a página "/sindico/aprovar-usuarios". Token JWT e dados do usuário são armazenados no cliente (localStorage).

**CT-003: Criação de Anúncio de Item por Usuário Aprovado** (Antigo CT-004)

* **Descrição:** Verificar se um usuário MORADOR logado e com status APROVADO consegue criar um novo anúncio do tipo ITEM.
* **Entrada (Modal de Criação de Anúncio):**
  + Título: "Monitor Gamer Usado"
  + Descrição: "Monitor 27 polegadas, 144Hz, em perfeito estado."
  + Tipo: "ITEM"
* **Procedimento:**
  + Navegar para a página de Anúncios.
  + Clicar em "+ Adicionar Novo Anúncio".
  + Preencher os dados do anúncio no modal.
  + Clicar em "Adicionar Anúncio".
* **Resultado Esperado:** Mensagem de sucesso "Anúncio adicionado com sucesso!" é exibida. O novo anúncio aparece na lista de anúncios do usuário e na lista geral de anúncios do seu condomínio. O anúncio é persistido no banco de dados.

**CT-004: Usuário Faz Oferta em Anúncio de Outro Usuário no Mesmo Condomínio** (Antigo CT-006)

* **Descrição:** Verificar se um usuário faz uma oferta em anúncio de outro usuário no mesmo condomínio.
* **Entrada (Modal de Fazer Oferta):**
  + Tipo de Oferta: "DINHEIRO"
  + Valor: 150.00
* **Procedimento:**
  + Usuário navega para a página de Anúncios.
  + Localiza o anúncio desejado.
  + Clica em "Fazer Oferta".
  + Preenche os dados da oferta no modal.
  + Clica em "Enviar Oferta".
* **Resultado Esperado:** Mensagem de sucesso "Oferta enviada com sucesso!" é exibida. A oferta é registrada no banco de dados com status "PENDENTE", associada ao anúncio e ao ofertante. O anunciante pode ver esta oferta em "Meus Anúncios".

**9. MATRIZES DE RASTREABILIDADE**

**9.1. Matriz de Rastreabilidade: Requisitos vs. Objetivos**

*Objetivos do Sistema (Exemplos, necessitam ser formalmente definidos para o projeto):*

* **OBJ-01:** Facilitar a troca e compartilhamento de itens e serviços entre vizinhos de um mesmo condomínio.
* **OBJ-02:** Fortalecer o senso de comunidade e colaboração dentro dos condomínios.
* **OBJ-03:** Proporcionar uma plataforma segura e controlada para interações condominiais.
* **OBJ-04:** Permitir administração centralizada de usuários e condomínios (para ADMINs).
* **OBJ-05:** Dar autonomia aos Síndicos para gerenciar os moradores de seus condomínios.
* Gráfico, Gráfico de linhas, Gráfico de dispersão

  O conteúdo gerado por IA pode estar incorreto.

**9.2. Matriz de Rastreabilidade: Requisitos vs. Casos de Teste**

Tabela

O conteúdo gerado por IA pode estar incorreto.

## 10. REFERÊNCIAS BIBLIOGRÁFICAS E FERRAMENTAS

**Tecnologias e Frameworks:**

Back-end:

* Java 17
* Spring Boot 3.4.5
* Spring Security
* Spring Data JPA
* Hibernate (via Spring Data JPA)
* java-jwt (Auth0) 4.5.0
* Maven (para gerenciamento de dependências e build)

Front-end:

* React 19.1.0
* React Router DOM 7.6.0
* Axios 1.9.0 (para requisições HTTP)
* Create React App (como base do projeto front-end)

Banco de Dados:

* Microsoft SQL Server

**Ferramentas de Desenvolvimento:**

* IDE Java: IntelliJ IDEA
* IDE JavaScript/React: VS Code
* Node.js e npm (para o ambiente de desenvolvimento front-end)
* Git (para controle de versão)
* Postman (para testes de API)
* Navegadores Web (Chrome, Firefox, etc., para desenvolvimento e teste do front-end)

**Outras Referências:**

* Princípios de Design de API REST.
* Padrões de Projeto de Software.
* Documentação oficial das tecnologias listadas.
* MODELO DE DOCUMENTAÇÃO DE REQUISITOS.docx.
* popula\_banco.py: Script Python para popular o banco de dados com dados de teste.