**Manual Definitivo para o Desenvolvedor Frontend - API de Gestão de Condomínios (v2.5)**

**Versão: 2.5 Data: 28 de junho de 2025**

**1. Visão Geral do Sistema**

**O Sistema de Gestão de Condomínios é uma plataforma de software robusta e centralizada, projetada para facilitar a administração e a vida em comunidade em condomínios residenciais, comerciais ou mistos. O objetivo é otimizar a comunicação, automatizar processos financeiros e gerenciar os recursos do condomínio de forma eficiente, transparente e acessível para todos os envolvidos.**

* **Tecnologia: Backend desenvolvido em Spring Boot (Java) com Spring Data JPA e Hibernate.**
* **Arquitetura: Aplicação monolítica expondo uma API RESTful.**
* **Comunicação: O formato de dados para todas as requisições e respostas é JSON.**
* **Modelo de API (Stateless): A API é stateless (sem estado). Cada chamada à API é independente e deve conter toda a informação necessária para ser processada, incluindo a autenticação via Token JWT.**

**2. Autenticação: A Porta de Entrada da API**

**Para usar qualquer funcionalidade protegida da API, o frontend precisa primeiro autenticar o usuário e obter um token de acesso. O sistema utiliza o padrão industrial JWT (JSON Web Tokens).**

* **Fluxo de Autenticação:**
  1. **O usuário insere e-mail e senha na tela de login.**
  2. **O frontend envia uma requisição POST para o endpoint /api/auth/login.**
  3. **Corpo da Requisição:**

**JSON**

**{**

**"email": "email.do.usuario@exemplo.com",**

**"senha": "senha\_do\_usuario"**

**}**

* 1. **Se as credenciais estiverem corretas, o backend retorna um status 200 OK com o token.**
  2. **O frontend deve armazenar este token de forma segura (ex: em memória ou no sessionStorage do navegador) e incluí-lo em todas as requisições subsequentes a endpoints protegidos no cabeçalho Authorization.**
  3. **Formato do Cabeçalho: Authorization: Bearer <token>. A palavra Bearer com um espaço é obrigatória.**
* **Expiração do Token: O token tem um tempo de expiração (atualmente 1 hora). Se a API retornar um erro 401 ou 403, o frontend deve tratar como uma sessão expirada e redirecionar para a tela de login.**

**3. Papéis de Usuário e Níveis de Permissão**

**O sistema possui uma hierarquia de papéis que define o que cada usuário pode ver e fazer (RBAC).**

* **Administrador Global: Acesso total e irrestrito.**
* **Gerente da Administradora: Acesso total aos condomínios gerenciados por sua empresa.**
* **Síndico: Autoridade máxima dentro de um único condomínio.**
* **Administrador do Condomínio: Funções administrativas em um único condomínio.**
* **Morador: Acesso restrito aos seus próprios dados e de suas unidades.**
* **Funcionário ADM / Porteiro: Papéis operacionais com permissões limitadas.**

**4. Referência Completa de Endpoints**

**Esta seção detalha cada recurso da API e como criar novos registros.**

* **Recurso: Pessoa (/api/pessoas)**
  + **POST /api/pessoas: Cadastra pessoas físicas ou jurídicas.**

**JSON**

**{**

**"pesNome": "Novo Morador João da Silva",**

**"pesCpfCnpj": "12345678901",**

**"pesTipo": "F",**

**"pesEmail": "joao.silva.novo@email.com",**

**"pesSenhaLogin": "senhaForte123"**

**}**

* + **Regra de Negócio Importante: O campo pesCpfCnpj passa por uma validação matemática de seus dígitos verificadores. Documentos inválidos serão rejeitados com um erro 400 Bad Request.**
* **Recurso: Administradora (/api/administradoras)**
  + **POST /api/administradoras: Cadastra uma empresa administradora.**

**JSON**

**{**

**"dadosEmpresa": { "pesCod": 2 },**

**"responsavel": { "pesCod": 3 }**

**}**

* **Recurso: Usuário da Administradora (/api/administradoras/usuarios)**
  + **POST /api/administradoras/usuarios: Vincula uma pessoa a uma administradora com um papel.**

**JSON**

**{**

**"administradora": { "admCod": 1 },**

**"pessoa": { "pesCod": 3 },**

**"aduPapel": "GERENTE"**

**}**

* **Recurso: Condomínio (/api/condominios)**
  + **POST /api/condominios: Cadastra um novo condomínio.**

**JSON**

**{**

**"conNome": "Residencial Jardins",**

**"conLogradouro": "Rua das Flores", "conNumero": "123",**

**"conTipologia": "RESIDENCIAL",**

**"administradora": { "admCod": 1 }**

**}**

* **Recurso: Unidade (/api/unidades)**
  + **POST /api/unidades: Cadastra uma nova unidade em um condomínio.**

**JSON**

**{**

**"uniNumero": "101-A",**

**"uniStatusOcupacao": "DESOCUPADO",**

**"uniValorTaxaCondominio": 750.50,**

**"condominio": { "conCod": 1 }**

**}**

* **Recurso: Morador (Vínculo) (/api/moradores)**
  + **POST /api/moradores: Associa uma pessoa a uma unidade.**

**JSON**

**{**

**"pessoa": { "pesCod": 4 },**

**"unidade": { "uniCod": 1 },**

**"morTipoRelacionamento": "PROPRIETARIO"**

**}**

* **Recurso: Usuário-Condomínio (Papel) (/api/usuarios/condominios)**
  + **POST /api/usuarios/condominios: Concede um papel a um usuário em um condomínio.**

**JSON**

**{**

**"pessoa": { "pesCod": 4 },**

**"condominio": { "conCod": 1 },**

**"uscPapel": "SINDICO"**

**}**

* **Recurso: Cobrança (/api/cobrancas)**
  + **POST /api/cobrancas: Cria uma cobrança individual.**
  + **POST /api/cobrancas/gerar-lote: Cria cobranças para todas as unidades de um condomínio.**
* **Recurso: Área Comum (/api/areascomuns)**
  + **POST /api/areascomuns: Cadastra uma nova área comum.**
* **Recurso: Solicitação de Manutenção (/api/manutencoes)**
  + **POST /api/manutencoes: Abre um chamado de manutenção.**

**5. Tratamento de Erros**

**A API sempre retornará um objeto JSON padronizado em caso de erro. O frontend deve utilizar o campo message para exibir o erro ao usuário.**

* **400 Bad Request: Dados enviados são inválidos. A message descreverá o erro específico.**
  + **Exemplo (Validação de CPF/CNPJ): {"message": "O CPF/CNPJ informado é inválido."}**
* **401 Unauthorized: E-mail/senha incorretos ou token inválido/ausente.**
* **403 Forbidden: Autenticado, mas sem permissão para a ação.**
* **404 Not Found: Recurso solicitado não existe.**
* **500 Internal Server Error: Erro inesperado no backend.**

**6. Processos Automatizados (Backend)**

**Existem rotinas no backend que são executadas automaticamente para manter a consistência dos dados. O frontend não interage diretamente com elas, mas pode observar seus resultados.**

* **Atualização de Status de Cobranças Vencidas:**
  + **O que faz? Uma rotina verifica todas as cobranças com status A\_VENCER. Se a data de vencimento for anterior ao dia atual, o status é alterado para VENCIDA.**
  + **Quando executa?**
    1. **Diariamente à meia-noite.**
    2. **Na inicialização do sistema.**
  + **Impacto para o Frontend: Uma cobrança pode mudar de status sem uma ação direta do usuário. A interface deve refletir corretamente o status atual (A\_VENCER, VENCIDA, PAGA, etc.) ao buscar os dados da API.**