**Manual Definitivo para o Desenvolvedor Frontend - API de Gestão de Condomínios (v2.4 - Consolidado)**

**Versão:** 2.4 **Data:** 18 de junho de 2025

**1. Visão Geral do Sistema**

**1.1. Qual a Utilidade 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.

**1.2. Como o Sistema Funciona?**

O sistema é composto por um backend (esta API) que contém todas as regras de negócio e o acesso ao banco de dados, e um frontend (a ser desenvolvido) que consumirá esta API para apresentar as informações e interagir com o usuário.

* **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). Isso significa que o servidor não guarda nenhuma informação sobre a sessão do usuário entre uma requisição e outra. 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.
   * **Corpo da Requisição:**

JSON

{

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

"senha": "senha\_do\_usuario"

}

1. O backend valida as credenciais. Se estiverem corretas, ele retorna um status 200 OK com o token.
   * **Corpo da Resposta:**

JSON

{

"token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJqYWNrc29ubmFzY2ltZW50bz..."

}

1. O frontend deve **armazenar este token** de forma segura (ex: em memória ou no sessionStorage do navegador).
2. Para **todas as requisições subsequentes** a endpoints protegidos, o frontend deve incluir o token no cabeçalho Authorization.
   * **IMPORTANTE:** O valor do cabeçalho deve seguir o formato Bearer <token>. A palavra **Bearer**, seguida de um **espaço**, é obrigatória.
   * **Exemplo de Cabeçalho:** Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJqYWNrc29ubmFzY2ltZW50bz...

**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).

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

**4. Referência Completa de Endpoints e Exemplos de Criação**

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"

}

**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: Tipo de Cobrança (/api/cobrancas/tipos)**

* **POST /api/cobrancas/tipos:** Cadastra um tipo de cobrança.

JSON

{

"ticDescricao": "Taxa Condominial Mensal"

}

**Recurso: Cobrança (/api/cobrancas)**

* **POST /api/cobrancas:** Cria uma cobrança individual.

JSON

{

"ficValorTaxa": 850.00,

"ficDtVencimento": "2025-07-10",

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

"tipoCobranca": { "ticCod": 1 }

}

* **POST /api/cobrancas/gerar-lote:** Cria cobranças para todas as unidades de um condomínio.
  + **Exemplo de URL:** POST .../gerar-lote?condominioId=1&dataVencimento=2025-07-10&tipoCobrancaId=1

**Recurso: Área Comum (/api/areascomuns)**

* **POST /api/areascomuns:** Cadastra uma nova área comum.

JSON

{

"arcNome": "Churrasqueira Gourmet",

"arcCapacidadeMaxima": 25,

"arcPermiteReserva": true,

"condominio": { "conCod": 1 }

}

**Recurso: Reserva de Área Comum (/api/reservas)**

* **POST /api/reservas:** Agenda uma reserva.

JSON

{

"dataHoraInicio": "2025-12-24T18:00:00",

"dataHoraFim": "2025-12-24T23:00:00",

"areaComum": { "arcCod": 1 },

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

"solicitante": { "pesCod": 4 }

}

**Recurso: Tipo de Solicitação de Manutenção (/api/manutencoes/tipos)**

* **POST /api/manutencoes/tipos:** Cadastra uma categoria de manutenção.

JSON

{

"tsmDescricao": "Elétrica"

}

**Recurso: Solicitação de Manutenção (/api/manutencoes)**

* **POST /api/manutencoes:** Abre um chamado de manutenção.

JSON

{

"localDescricao": "Luz do corredor do 10º andar piscando.",

"descricaoProblema": "A lâmpada está piscando sem parar.",

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

"solicitante": { "pesCod": 4 },

"tipoSolicitacao": { "tsmCod": 1 }

}

**Recurso: Comunicação (Comunicados)**

* **Fluxo:** Primeiro cria-se o comunicado (/api/comunicacoes). Se o destino for para pessoas específicas, faz-se chamadas para /api/comunicados/entregas.
* **A. Criar Comunicado (POST /api/comunicacoes):**

JSON

{

"comAssunto": "Manutenção da Piscina",

"comMensagem": "A piscina estará fechada na próxima segunda-feira.",

"comDesTodos": "ESPECIFICOS",

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

"remetente": { "pesCod": 4 }

}

* **B. Registrar Entrega Específica (POST /api/comunicados/entregas):**

JSON

{

"comunicado": { "comCod": 1 },

"destinatario": { "pesCod": 5 }

}

**Recurso: Assembleia**

* **Fluxo:** Crie a Assembleia, depois seus Tópicos, adicione Participantes e registre os Votos.
* **A. Criar Assembleia (POST /api/assembleias):**

JSON

{

"assDescricao": "Assembleia Geral Ordinária",

"assDataHora": "2025-08-15T20:00:00",

"condominio": { "conCod": 1 }

}

* **B. Adicionar Tópico (POST /api/assembleias/topicos):**

JSON

{

"aspDescricao": "Votação sobre a nova pintura da fachada",

"assembleia": { "assCod": 1 }

}

* **C. Adicionar Participante (POST /api/assembleias/participantes):**

JSON

{

"assembleia": { "assCod": 1 },

"pessoa": { "pesCod": 4 }

}

* **D. Registrar Voto (POST /api/assembleias/votos):**

JSON

{

"topico": { "aspCod": 1 },

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

"voto": "SIM"

}

**Recurso: Documento**

* **Fluxo:** Faça o upload do arquivo no seu storage, depois crie o registro do documento na API. Se a permissão for para pessoas selecionadas, adicione as permissões em seguida.
* **A. Criar Documento (POST /api/documentos):**

JSON

{

"nome": "Ata da Assembleia de Agosto 2025",

"tipo": "application/pdf",

"caminhoArquivo": "/caminho/no/storage/ata\_agosto\_2025.pdf",

"permissaoVisualizar": "S",

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

"uploader": { "pesCod": 4 }

}

* **B. Dar Permissão Específica (POST /api/documentos/permissoes):**

JSON

{

"documento": { "docCod": 1 },

"pessoa": { "pesCod": 5 }

}

**5. Tratamento de Erros**

A API sempre retornará um objeto JSON padronizado em caso de erro.

* **400 Bad Request:** Dados enviados inválidos. A message descreverá o erro.
* **401 Unauthorized:** E-mail/senha incorretos.
* **403 Forbidden:** Autenticado, mas sem permissão para a ação.
* **404 Not Found:** Recurso não existe.
* **500 Internal Server Error:** Erro inesperado no backend.
* **Exemplo de Resposta de Erro:**

JSON

{

"timestamp": "2025-06-18T08:10:00.123456",

"status": 403,

"error": "Forbidden",

"message": "Acesso negado. Você não tem permissão para executar esta ação.",

"path": "/api/pessoas/1/inativar"

}