

# Documentação da API - Cronograma UBSF

---

## Base URL

**Local:** <http://localhost:3000>

**Produção:** <https://seu-projeto.vercel.app>

---

## Autenticação

A API utiliza autenticação JWT (JSON Web Tokens). Após o login ou cadastro, você receberá um token que deve ser incluído no header **Authorization** de suas requisições protegidas.

### Formato do Header:

```
Authorization: Bearer SEU_TOKEN_AQUI
```

---

## Estrutura de Resposta Padrão

Todas as respostas seguem o mesmo formato:

```
{
  "success": boolean,
  "message": "string",
  "data": object | array | null,
  "timestamp": "ISO 8601 string"
}
```

## Códigos de Status HTTP

- **200** - Sucesso
  - **201** - Criado com sucesso
  - **400** - Dados inválidos
  - **401** - Não autorizado
  - **403** - Sem permissão
  - **404** - Não encontrado
  - **405** - Método não permitido
  - **409** - Conflito (duplicata)
  - **500** - Erro interno do servidor
- 

## Endpoints da API

## 1. Health Check

### GET /api/health

Verifica o status da API e conexão com o banco de dados.

#### Response:

```
{
  "success": true,
  "message": "API está funcionando corretamente",
  "data": {
    "status": "OK",
    "timestamp": "2024-01-15T10:30:00.000Z",
    "version": "1.0.0",
    "database": "Connected",
    "environment": "development"
  },
  "timestamp": "2024-01-15T10:30:00.000Z"
}
```

---

## 2. Autenticação

### POST /api/auth/cadastro

Cadastra um novo usuário no sistema.

#### Request Body:

```
{
  "nome": "Maria Silva Santos",
  "email": "maria@email.com",
  "senha": "minhasenha123",
  "cargo": "enfermeiro"
}
```

#### Exemplo de Request:

```
// JavaScript/React Native
const response = await fetch('/api/auth/cadastro', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
  },
  body: JSON.stringify({
    nome: "Maria Silva Santos",
```

```
        email: "maria@email.com",
        senha: "minhasenha123",
        cargo: "enfermeiro"
    })
});
const data = await response.json();
```

### Response (201):

```
{
  "success": true,
  "message": "Usuário cadastrado com sucesso",
  "data": {
    "usuario": {
      "id": "cm123abc",
      "email": "maria@email.com",
      "nome": "Maria Silva Santos",
      "cargo": "enfermeiro",
      "criadoEm": "2024-01-15T10:30:00.000Z",
      "atualizadoEm": "2024-01-15T10:30:00.000Z"
    },
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
  },
  "timestamp": "2024-01-15T10:30:00.000Z"
}
```

### POST /api/auth/login

Realiza login no sistema.

### Request Body:

```
{
  "email": "maria@email.com",
  "senha": "minhasenha123"
}
```

### Response (200):

```
{
  "success": true,
  "message": "Login realizado com sucesso",
  "data": {
    "usuario": {
      "id": "cm123abc",
      "email": "maria@email.com",
```

```
{
  "nome": "Maria Silva Santos",
  "cargo": "enfermeiro",
  "criadoEm": "2024-01-15T10:30:00.000Z",
  "atualizadoEm": "2024-01-15T10:30:00.000Z"
},
{
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
},
{
  "timestamp": "2024-01-15T10:30:00.000Z"
}
}
```

---

### 3. Cronogramas

#### GET /api/cronogramas

Lista todos os cronogramas com paginação.

#### Query Parameters:

- **page** (opcional): Número da página (padrão: 1)
- **limit** (opcional): Itens por página (padrão: 10, máximo: 50)
- **mes** (opcional): Filtrar por mês (1-12)
- **ano** (opcional): Filtrar por ano

#### Exemplo de Request:

```
// JavaScript/React Native
const response = await fetch('/api/cronogramas?page=1&limit=10&mes=1&ano=2024');
const data = await response.json();
```

#### Response:

```
{
  "success": true,
  "message": "Cronogramas listados com sucesso",
  "data": {
    "cronogramas": [
      {
        "id": "cm123abc",
        "mes": 1,
        "ano": 2024,
        "nomeUBSF": "UBSF Centro",
        "enfermeiro": "Maria Silva Santos",
        "medico": "Dr. João Carlos Oliveira",
        "criadoEm": "2024-01-01T00:00:00.000Z",
        "atualizadoEm": "2024-01-01T00:00:00.000Z",
        "atividades": [
```

```

    {
      "id": "cm456def",
      "cronogramaId": "cm123abc",
      "data": "2024-01-02T00:00:00.000Z",
      "diaSemana": "TERÇA-MANHÃ",
      "descricao": "Consultas de rotina - Hipertensão e Diabetes",
      "criadoEm": "2024-01-01T00:00:00.000Z"
    }
  ]
},
"pagination": {
  "page": 1,
  "limit": 10,
  "total": 25,
  "totalPages": 3
}
},
"timestamp": "2024-01-15T10:30:00.000Z"
}

```

## POST /api/cronogramas

Cria um novo cronograma.

### Request Body:

```

{
  "mes": 3,
  "ano": 2024,
  "nomeUBSF": "UBSF Vila Nova",
  "enfermeiro": "Ana Paula Costa",
  "medico": "Dra. Fernanda Lima"
}

```

### Exemplo de Request:

```

// JavaScript/React Native
const response = await fetch('/api/cronogramas', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
  },
  body: JSON.stringify({
    mes: 3,
    ano: 2024,
    nomeUBSF: "UBSF Vila Nova",
    enfermeiro: "Ana Paula Costa",

```

```
    medico: "Dra. Fernanda Lima"
  })
});
const data = await response.json();
```

#### Response (201):

```
{
  "success": true,
  "message": "Cronograma criado com sucesso",
  "data": {
    "id": "cm789ghi",
    "mes": 3,
    "ano": 2024,
    "nomeUBSF": "UBSF Vila Nova",
    "enfermeiro": "Ana Paula Costa",
    "medico": "Dra. Fernanda Lima",
    "criadoEm": "2024-01-15T10:30:00.000Z",
    "atualizadoEm": "2024-01-15T10:30:00.000Z"
  },
  "timestamp": "2024-01-15T10:30:00.000Z"
}
```

---

## 4. Cronograma Específico

### GET /api/cronogramas/[id]

Busca um cronograma específico por ID.

#### Exemplo de Request:

```
const response = await fetch('/api/cronogramas/cm123abc');
const data = await response.json();
```

#### Response:

```
{
  "success": true,
  "message": "Cronograma encontrado",
  "data": {
    "id": "cm123abc",
    "mes": 1,
    "ano": 2024,
    "nomeUBSF": "UBSF Centro",
    "enfermeiro": "Maria Silva Santos",
  }
}
```

```
{
  "medico": "Dr. João Carlos Oliveira",
  "criadoEm": "2024-01-01T00:00:00.000Z",
  "atualizadoEm": "2024-01-01T00:00:00.000Z",
  "atividades": [
    {
      "id": "cm456def",
      "cronogramaId": "cm123abc",
      "data": "2024-01-02T00:00:00.000Z",
      "diaSemana": "TERÇA-MANHÃ",
      "descricao": "Consultas de rotina",
      "criadoEm": "2024-01-01T00:00:00.000Z"
    }
  ]
},
"timestamp": "2024-01-15T10:30:00.000Z"
}
```

### PUT /api/cronogramas/[id]

Atualiza um cronograma existente.

#### Request Body:

```
{
  "nomeUBSF": "UBSF Centro Atualizado",
  "enfermeiro": "Maria Silva Santos",
  "medico": "Dr. João Carlos Oliveira"
}
```

#### Exemplo de Request:

```
const response = await fetch('/api/cronogramas/cm123abc', {
  method: 'PUT',
  headers: {
    'Content-Type': 'application/json',
  },
  body: JSON.stringify({
    nomeUBSF: "UBSF Centro Atualizado",
    enfermeiro: "Maria Silva Santos",
    medico: "Dr. João Carlos Oliveira"
  })
});
```

### DELETE /api/cronogramas/[id]

Deleta um cronograma e todas suas atividades.

### Exemplo de Request:

```
const response = await fetch('/api/cronogramas/cm123abc', {
  method: 'DELETE'
});
```

---

## 5. Atividades do Cronograma

### GET /api/cronogramas/[id]/atividades

Lista atividades de um cronograma específico.

#### Query Parameters:

- **page** (opcional): Número da página
- **limit** (opcional): Itens por página
- **diaSemana** (opcional): Filtrar por dia da semana
- **dataInicio** (opcional): Data inicial (YYYY-MM-DD)
- **dataFim** (opcional): Data final (YYYY-MM-DD)

### Exemplo de Request:

```
const response = await fetch('/api/cronogramas/cm123abc/atividades?
diaSemana=SEGUNDA-MANHÃ');
```

### POST /api/cronogramas/[id]/atividades

Cria uma nova atividade para o cronograma.

#### Request Body:

```
{
  "data": "2024-01-15",
  "diaSemana": "SEGUNDA-MANHÃ",
  "descricao": "Consultas de rotina - Hipertensão"
}
```

### Exemplo de Request:

```
const response = await fetch('/api/cronogramas/cm123abc/atividades', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
```



```
    },  
    body: JSON.stringify({  
      data: "2024-01-15",  
      diaSemana: "SEGUNDA-MANHÃ",  
      descricao: "Consultas de rotina - Hipertensão"  
    })  
  });
```

---

## 6. Atividade Específica

### GET /api/atividades/[id]

Busca uma atividade específica por ID.

### PUT /api/atividades/[id]

Atualiza uma atividade existente.

#### Request Body:

```
{  
  "data": "2024-01-16",  
  "diaSemana": "TERÇA-MANHÃ",  
  "descricao": "Consultas de rotina - Diabetes"  
}
```

### DELETE /api/atividades/[id]

Deleta uma atividade específica.



## Valores Válidos

### Dias da Semana

- SEGUNDA-MANHÃ
- SEGUNDA-TARDE
- TERÇA-MANHÃ
- TERÇA-TARDE
- QUARTA-MANHÃ
- QUARTA-TARDE
- QUINTA-MANHÃ
- QUINTA-TARDE
- SEXTA-MANHÃ
- SEXTA-TARDE
- SABADO-MANHÃ

- SABADO-TARDE

## Meses

- Valores de 1 a 12 (Janeiro = 1, Dezembro = 12)

## Cargos de Usuário

- enfermeiro
- medico
- admin

---

## ✕ Tratamento de Erros

### Erro de Validação (400)

```
{
  "success": false,
  "message": "Dados inválidos: \"mes\" is required",
  "data": null,
  "timestamp": "2024-01-15T10:30:00.000Z"
}
```

### Não Encontrado (404)

```
{
  "success": false,
  "message": "Cronograma não encontrado",
  "data": null,
  "timestamp": "2024-01-15T10:30:00.000Z"
}
```

### Conflito - Duplicata (409)

```
{
  "success": false,
  "message": "Já existe um cronograma para este mês/ano",
  "data": null,
  "timestamp": "2024-01-15T10:30:00.000Z"
}
```

---

## Exemplos de Uso no Frontend

```
// services/api.js
const API_BASE_URL = __DEV__
  ? 'http://localhost:3000'
  : 'https://seu-projeto.vercel.app';

export const api = {
  // Autenticação
  async cadastrar(userData) {
    const response = await fetch(`${API_BASE_URL}/api/auth/cadastro`, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(userData),
    });
    return response.json();
  },

  async login(credentials) {
    const response = await fetch(`${API_BASE_URL}/api/auth/login`, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(credentials),
    });
    return response.json();
  },

  // Listar cronogramas
  async getCronogramas(params = {}) {
    const queryString = new URLSearchParams(params).toString();
    const response = await fetch(`${API_BASE_URL}/api/cronogramas?${queryString}`);
    return response.json();
  },

  // Criar cronograma
  async createCronograma(data) {
    const response = await fetch(`${API_BASE_URL}/api/cronogramas`, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify(data),
    });
    return response.json();
  },
}
```

```

// Buscar cronograma por ID
async getCronograma(id) {
  const response = await fetch(`${API_BASE_URL}/api/cronogramas/${id}`);
  return response.json();
},

// Atualizar cronograma
async updateCronograma(id, data) {
  const response = await fetch(`${API_BASE_URL}/api/cronogramas/${id}`, {
    method: 'PUT',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify(data),
  });
  return response.json();
},

// Deletar cronograma
async deleteCronograma(id) {
  const response = await fetch(`${API_BASE_URL}/api/cronogramas/${id}`, {
    method: 'DELETE',
  });
  return response.json();
},

// Listar atividades do cronograma
async getAtividades(cronogramaId, params = {}) {
  const queryString = new URLSearchParams(params).toString();
  const response = await
fetch(`${API_BASE_URL}/api/cronogramas/${cronogramaId}/atividades?
${queryString}`);
  return response.json();
},

// Criar atividade
async createAtividade(cronogramaId, data) {
  const response = await
fetch(`${API_BASE_URL}/api/cronogramas/${cronogramaId}/atividades`, {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json',
    },
    body: JSON.stringify(data),
  });
  return response.json();
},

// Atualizar atividade
async updateAtividade(id, data) {
  const response = await fetch(`${API_BASE_URL}/api/atividades/${id}`, {
    method: 'PUT',
    headers: {

```

```

        'Content-Type': 'application/json',
      },
      body: JSON.stringify(data),
    });
    return response.json();
  },

  // Deletar atividade
  async deleteAtividade(id) {
    const response = await fetch(`${API_BASE_URL}/api/atividades/${id}`, {
      method: 'DELETE',
    });
    return response.json();
  }
};

```

## Exemplo de Componente React Native

```

// components/CronogramaList.js
import React, { useState, useEffect } from 'react';
import { View, Text, FlatList, TouchableOpacity } from 'react-native';
import { api } from '../services/api';

export default function CronogramaList() {
  const [cronogramas, setCronogramas] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

  useEffect(() => {
    loadCronogramas();
  }, []);

  const loadCronogramas = async () => {
    try {
      setLoading(true);
      const response = await api.getCronogramas({ limit: 20 });

      if (response.success) {
        setCronogramas(response.data.cronogramas);
      } else {
        setError(response.message);
      }
    } catch (err) {
      setError('Erro ao carregar cronogramas');
    } finally {
      setLoading(false);
    }
  };

  const renderCronograma = ({ item }) => (

```

```

    <TouchableOpacity style={styles.cronogramaItem}>
      <Text style={styles.title}>
        {item.nomeUBSF} - {item.mes}/{item.ano}
      </Text>
      <Text>Enfermeiro: {item.enfermeiro}</Text>
      <Text>Médico: {item.medico}</Text>
      <Text>Atividades: {item.atividades.length}</Text>
    </TouchableOpacity>
  );

  if (loading) return <Text>Carregando...</Text>;
  if (error) return <Text>Erro: {error}</Text>;

  return (
    <FlatList
      data={cronogramas}
      renderItem={renderCronograma}
      keyExtractor={(item) => item.id}
      onRefresh={loadCronogramas}
      refreshing={loading}
    />
  );
}

```



## Dicas de Performance

1. **Paginação:** Sempre use paginação para listas grandes
2. **Cache:** Implemente cache local para dados que não mudam frequentemente
3. **Loading States:** Sempre mostre estados de carregamento
4. **Error Handling:** Trate todos os possíveis erros da API
5. **Retry Logic:** Implemente retry automático para falhas de rede



## Configuração para Desenvolvimento

### Android (Emulador)

```
const API_BASE_URL = 'http://10.0.2.2:3000'; // Para emulador Android
```

### iOS (Simulador)

```
const API_BASE_URL = 'http://localhost:3000'; // Para simulador iOS
```

### Dispositivo Físico

```
const API_BASE_URL = 'http://SEU_IP_LOCAL:3000'; // Ex:  
http://192.168.1.100:3000
```

---

## Suporte

Para dúvidas ou problemas com a API, consulte:

- Logs do servidor para debugging
- Health check endpoint para verificar status
- Esta documentação para referência completa

**Última atualização:** Janeiro 2024