Documentação Técnica - API do Sistema de Agendamentos

Visão Geral

Esta API REST foi desenvolvida em Flask e fornece todas as funcionalidades necessárias para o sistema de agendamentos de salões de beleza e barbearias.

Tecnologias Utilizadas

- **Framework**: Flask 2.3+
- **Banco de Dados**: SQLite (desenvolvimento) / PostgreSQL (produção)
- **ORM**: SQLAlchemy
- **Autenticação**: JWT (JSON Web Tokens)
- **Validação**: Flask-WTF
- **CORS**: Flask-CORS

URL Base

```
http://localhost:5000/api/v1
```

Autenticação

A API utiliza JWT (JSON Web Tokens) para autenticação. Todas as rotas protegidas requerem o header:

```
Authorization: Bearer <token>
```

Obter Token

POST `/auth/login`

```
{
  "email": "usuario@email.com",
```

```
  "password": "senha123"
}
```

Resposta de Sucesso (200):

```
{
  "success": true,
  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9...",
  "user": {
    "id": 1,
    "name": "João Silva",
    "email": "joao@email.com",
    "user_type": "client"
  }
}
```

Endpoints da API

Usuários

Criar Usuário

POST `/users`

```
{
  "name": "João Silva",
  "email": "joao@email.com",
  "password": "senha123",
  "phone": "(11) 99999-9999",
  "user_type": "client",
  "business_name": "Salão Beleza Total",
  "address": "Rua das Flores, 123",
  "description": "Especializado em cortes modernos"
}
```

Obter Usuário

GET `/users/<id>`

Resposta:

```
{
  "id": 1,
  "name": "João Silva",
```

```
  "email": "joao@email.com",
  "phone": "(11) 99999-9999",
  "user_type": "provider",
  "business_name": "Salão Beleza Total",
  "address": "Rua das Flores, 123",
  "description": "Especializado em cortes modernos",
  "created_at": "2024-01-01T10:00:00Z"
}
```

Atualizar Usuário

PUT `/users/<id>`

Listar Usuários

GET `/users`

Parâmetros de query: - `user_type`: Filtrar por tipo (client/provider) - `page`: Número da página (padrão: 1) - `per_page`: Itens por página (padrão: 10)

Serviços

Criar Serviço

POST `/services`

```
{
  "name": "Corte Masculino",
  "description": "Corte moderno com acabamento",
  "price": 35.00,
  "duration": 30,
  "category": "Cortes",
  "active": true
}
```

Listar Serviços

GET `/services`

Parâmetros: - `provider_id`: Filtrar por prestador - `category`: Filtrar por categoria - `active`: Filtrar por status (true/false)

Resposta:

```
{
  "services": [
    {
      "id": 1,
      "name": "Corte Masculino",
      "description": "Corte moderno com acabamento",
      "price": 35.00,
      "duration": 30,
      "category": "Cortes",
      "active": true,
      "provider_id": 2,
      "provider_name": "João Silva"
    }
  ],
  "total": 1,
  "page": 1,
  "per_page": 10
}
```


Obter Serviço

GET `/services/<id>`

Atualizar Serviço

PUT `/services/<id>`

Deletar Serviço

DELETE `/services/<id>`

Agendamentos

Criar Agendamento

POST `/appointments`


```
{
  "service_id": 1,
  "provider_id": 2,
  "appointment_date": "2024-01-15T14:30:00Z",
  "notes": "Observações especiais"
}
```

Listar Agendamentos

GET `/appointments`

Parâmetros: - `client_id`: Filtrar por cliente - `provider_id`: Filtrar por prestador - `status`: Filtrar por status - `date_from`: Data inicial (YYYY-MM-DD) - `date_to`: Data final (YYYY-MM-DD)

Resposta:

```
{
  "appointments": [
    {
      "id": 1,
      "service_id": 1,
      "service_name": "Corte Masculino",
      "service_price": 35.00,
      "client_id": 1,
      "client_name": "Maria Santos",
      "provider_id": 2,
      "provider_name": "João Silva",
      "appointment_date": "2024-01-15T14:30:00Z",
      "status": "confirmed",
      "notes": "Observações especiais",
      "created_at": "2024-01-10T10:00:00Z"
    }
  ],
  "total": 1
}
```

Obter Agendamento

GET `/appointments/<id>`

Atualizar Status do Agendamento

PUT `/appointments/<id>/status`

```
{
  "status": "confirmed",
  "notes": "Agendamento confirmado"
}
```

Status válidos: - `pending`: Pendente - `confirmed`: Confirmado - `in_progress`: Em andamento - `completed`: Concluído - `cancelled`: Cancelado - `no_show`: Não compareceu

Cancelar Agendamento

DELETE `/appointments/<id>`

Códigos de Status HTTP

- **200**: Sucesso
- **201**: Criado com sucesso
- **400**: Erro de validação
- **401**: Não autorizado
- **403**: Acesso negado
- **404**: Não encontrado
- **409**: Conflito (ex: horário já ocupado)
- **500**: Erro interno do servidor

Tratamento de Erros

Todas as respostas de erro seguem o padrão:

```
{
  "success": false,
  "message": "Descrição do erro",
  "errors": {
    "campo": ["Lista de erros específicos"]
  }
}
```

Exemplos de Erros

Erro de Validação (400):

```
{
  "success": false,
  "message": "Dados inválidos",
  "errors": {
    "email": ["Email é obrigatório"],
    "password": ["Senha deve ter pelo menos 6 caracteres"]
  }
}
```

Erro de Autenticação (401):

```
{
  "success": false,
  "message": "Token inválido ou expirado"
}
```

Configuração e Deploy

Variáveis de Ambiente

```
# Configuração do Flask
FLASK_ENV=production
SECRET_KEY=sua_chave_secreta_muito_segura

# Banco de Dados
DATABASE_URL=sqlite:///app.db

# JWT
JWT_SECRET_KEY=sua_chave_jwt_secreta
JWT_ACCESS_TOKEN_EXPIRES=3600

# CORS
CORS_ORIGINS=http://localhost:3000,https://seudominio.com
```

Instalação

```
# Clonar repositório
git clone <repositorio>
cd salon-booking-api

# Criar ambiente virtual
python -m venv venv
source venv/bin/activate  # Linux/Mac
# ou
venv\Scripts\activate  # Windows

# Instalar dependências
pip install -r requirements.txt

# Configurar banco de dados
flask db upgrade

# Executar aplicação
python src/main.py
```

Docker

```dockerfile
FROM python:3.11-slim

WORKDIR /app
COPY requirements.txt .
RUN pip install -r requirements.txt

COPY src/ ./src/
EXPOSE 5000

CMD ["python", "src/main.py"]
```

Testes

Executar Testes

```bash
# Testes unitários
python -m pytest tests/

# Testes com cobertura
python -m pytest --cov=src tests/

# Testes de integração
python -m pytest tests/integration/
```

Exemplo de Teste

```python
def test_create_user():
    response = client.post('/users', json={
        'name': 'Teste',
        'email': 'teste@email.com',
        'password': 'senha123',
        'user_type': 'client'
    })

    assert response.status_code == 201
    assert response.json['success'] == True
```

Monitoramento

Logs

A aplicação gera logs estruturados em JSON:

```
{
  "timestamp": "2024-01-01T10:00:00Z",
  "level": "INFO",
  "message": "User created successfully",
  "user_id": 1,
  "request_id": "abc123"
}
```

Métricas

- Tempo de resposta das APIs
- Taxa de erro por endpoint
- Número de usuários ativos
- Agendamentos por período

Segurança

Medidas Implementadas

- **Autenticação JWT**: Tokens com expiração
- **Validação de Entrada**: Sanitização de dados
- **CORS Configurado**: Apenas origens autorizadas
- **Rate Limiting**: Limite de requisições por IP
- **HTTPS**: Obrigatório em produção

Boas Práticas

- Senhas são hasheadas com bcrypt
- Tokens JWT têm tempo de expiração
- Logs não contêm informações sensíveis
- Validação rigorosa de entrada de dados

Versão da API: 1.0
Última Atualização: Dezembro 2024