Documentação Técnica - Sistema de Agendamento

Sumário

1. Visão Geral
2. Arquitetura do Sistema
3. Backend - Informações Técnicas
4. Frontend - Informações Técnicas
5. Regras de Negócio
6. Estrutura de Dados
7. Fluxos Principais
8. Segurança e Autenticação
9. API e Endpoints

Visão Geral

O Sistema de Agendamento é uma aplicação web desenvolvida para conectar clientes (CLIENTE) e prestadores de serviço (PRESTADOR), permitindo que clientes agendem serviços oferecidos por lojas de prestadores.

Objetivo

Facilitar o agendamento de serviços entre clientes e prestadores, fornecendo uma plataforma completa que inclui:

- Cadastro e autenticação de usuários
- Gerenciamento de lojas e serviços
- Sistema de agendamento com validação de horários
- Interface responsiva para web e mobile

Tipos de Usuário

O sistema suporta dois tipos de usuários:

1. **CLIENTE**: Usuários que buscam e agendam serviços
2. **PRESTADOR**: Usuários que possuem lojas e oferecem serviços

---

Arquitetura do Sistema

Arquitetura Geral

```
│   MySQL (Docker)│
└─────────────────┘
```

Padrão de Arquitetura

- **Backend**: Arquitetura em camadas (Controller → Service → Repository)
- **Frontend**: Arquitetura baseada em componentes (Component → Service)
- **Comunicação**: REST API com autenticação JWT
- **Banco de Dados**: Relacional (MySQL) com TypeORM

Backend - Informações Técnicas

Stack Tecnológica

Tecnologia	Versão	Propósito
Node.js	v18+	Runtime JavaScript
TypeScript	5.1.3	Linguagem de programação
NestJS	10.0.0	Framework Node.js
TypeORM	0.3.17	ORM para banco de dados
MySQL	8.0	Banco de dados relacional
Docker	-	Containerização
bcrypt	5.1.1	Hash de senhas
JWT	10.2.0	Autenticação
class-validator	0.14.0	Validação de DTOs
class-transformer	0.5.1	Transformação de objetos

Estrutura de Módulos

```
backend-tcc/
├── src/
│   ├── appointment/      # Módulo de agendamentos
│   ├── auth/            # Módulo de autenticação
│   ├── core/            # Configurações globais
│   │   ├── guards/      # Guards de autenticação
│   │   ├── interceptors/# Interceptors
│   │   └── decorators/  # Decorators customizados
│   ├── file/            # Módulo de upload de arquivos
│   ├── service/         # Módulo de serviços
│   ├── store/           # Módulo de lojas
│   ├── user/            # Módulo de usuários
│   ├── app.module.ts    # Módulo raiz
│   └── main.ts          # Entry point
├── test/                # Testes
```

```
├── docker-compose.yml    # Configuração Docker
└── package.json          # Dependências
```

Princípios de Desenvolvimento

- **SOLID**: Aplicação dos princípios SOLID
- **Clean Architecture**: Separação de responsabilidades
- **DRY**: Evitar repetição de código
- **Type Safety**: Uso rigoroso de TypeScript
- **Validação**: Validação de dados em todas as camadas

Configuração do Banco de Dados

- **ORM**: TypeORM
- **Banco**: MySQL 8.0
- **Containerização**: Docker Compose
- **Migrations**: TypeORM Migrations
- **Sincronização**: Desabilitada em produção

Frontend - Informações Técnicas

Stack Tecnológica

Tecnologia	Versão	Propósito
Angular	20.0.0	Framework frontend
Ionic	8.0.0	Framework mobile
TypeScript	5.8.0	Linguagem de programação
Capacitor	7.4.4	Bridge nativo
RxJS	7.8.0	Programação reativa
SCSS	-	Pré-processador CSS

Estrutura de Módulos

```
agendoo-front/
├── src/
│   ├── app/
│   │   ├── cliente/          # Módulo do cliente
│   │   │   ├── busca/        # Busca de lojas
│   │   │   ├── agendar/       # Agendamento
│   │   │   ├── agendamentos/  # Lista de agendamentos
│   │   │   └── perfil/        # Perfil do cliente
│   │   ├── prestador/         # Módulo do prestador
│   │   │   ├── home/          # Dashboard
│   │   │   ├── servicos/      # Gerenciamento de serviços
│   │   │   ├── configuracoes/ # Configurações da loja
│   │   │   └── perfil/        # Perfil do prestador
│   │   ├── login/             # Tela de login
```

```
│   │   ├── register/          # Tela de registro
│   │   ├── services/          # Serviços HTTP
│   │   ├── models/            # Modelos de dados
│   │   └── interceptors/      # Interceptors HTTP
│   ├── assets/                # Recursos estáticos
│   └── environments/          # Configurações de ambiente
└── android/                   # Build Android (Capacitor)
```

Características

- **Responsivo**: Adaptável a diferentes tamanhos de tela
- **Mobile-First**: Otimizado para dispositivos móveis
- **PWA Ready**: Preparado para Progressive Web App
- **Cross-Platform**: Suporta Android via Capacitor

Regras de Negócio

1. Módulo de Usuários

1.1 Tipos de Usuário

- **CLIENTE**: Deve possuir telefone obrigatório
- **PRESTADOR**: Deve possuir CPF obrigatório

1.2 Validações

- Email deve ser único no sistema
- Senha mínima de 6 caracteres
- Nome mínimo de 2 caracteres
- CPF no formato XXX.XXX.XXX-XX
- Telefone no formato válido (10-14 dígitos)

1.3 Segurança

- Senhas são hasheadas com bcrypt (10 rounds)
- Senhas nunca são retornadas em respostas da API
- IDs são UUIDs para maior segurança

2. Módulo de Autenticação

2.1 Signup (Cadastro)

- Cria novo usuário e retorna token JWT
- Valida tipo de usuário e campos obrigatórios
- Email deve ser único (erro 409 se duplicado)

2.2 Login

- Autentica com email e senha
- Retorna token JWT válido por 24 horas (configurável)
- Mensagem genérica "Invalid credentials" para prevenir enumeração

2.3 Reset de Senha

- Suporta três métodos:
    1. Request reset: Solicita código por email
    2. Confirm reset: Confirma com código

3. Reset password: Redefine diretamente com email

3. Módulo de Lojas

3.1 Regras de Negócio

- Apenas usuários PRESTADOR podem ter lojas
- Cada PRESTADOR pode ter apenas uma loja
- Tentativa de criar segunda loja resulta em erro 409

3.2 Horários de Funcionamento

- Deve incluir todos os 7 dias da semana (0-6)
- Cada dia deve ser único (sem duplicatas)
- Se `isOpen = true`, `openTime` e `closeTime` são obrigatórios
- Formato de horário: HH:mm (24 horas)

3.3 Localização

- Campos obrigatórios: rua, número, bairro, cidade, estado, CEP
- Estado: 2 caracteres (abreviação)
- CEP: formato XXXXX-XXX ou XXXXXXXX
- Coordenadas (latitude/longitude) opcionais

3.4 Intervalo de Agendamento

- Configurável por loja: 5, 10, 15 ou 30 minutos
- Define a granularidade dos horários disponíveis

4. Módulo de Serviços

4.1 Regras de Negócio

- Serviços pertencem a uma loja
- Apenas o dono da loja pode criar/editar/deletar serviços
- Serviços são públicos (qualquer um pode visualizar)

4.2 Validações

- Título: mínimo 2 caracteres
- Descrição: mínimo 10 caracteres
- Preço: mínimo R$ 0,01
- Duração: 1 a 1440 minutos (24 horas)

4.3 Autorização

- Criação: automaticamente associado à loja do usuário autenticado
- Edição/Exclusão: apenas dono da loja (erro 403 se não autorizado)

5. Módulo de Agendamentos

5.1 Regras de Negócio

- Apenas usuários CLIENTE podem criar agendamentos
- Agendamentos são criados com status PENDING
- Data do agendamento deve ser futura
- Horário deve estar dentro do horário de funcionamento da loja

5.2 Validação de Conflitos

O sistema valida conflitos considerando:

- Horário de funcionamento da loja
- Intervalo de agendamento da loja
- Duração do serviço
- Agendamentos existentes (exceto cancelados)

Algoritmo de Validação:

1. Verifica se a loja está aberta no dia/horário
2. Verifica se o horário não conflita com agendamentos existentes
3. Considera a duração do serviço para detectar sobreposições
4. Agendamentos cancelados são ignorados na verificação

5.3 Status de Agendamento

- **PENDING**: Pendente de confirmação (padrão)
- **CONFIRMED**: Confirmado
- **COMPLETED**: Concluído
- **CANCELLED**: Cancelado

5.4 Regras de Status

- Agendamentos cancelados não podem ser atualizados ou cancelados novamente
- Agendamentos concluídos não podem ser cancelados ou atualizados
- Apenas o dono do agendamento pode visualizar/editar/cancelar

5.5 Horários Disponíveis

O sistema calcula horários disponíveis considerando:

- Horário de funcionamento da loja
- Intervalo de agendamento (5, 10, 15 ou 30 min)
- Duração do serviço
- Agendamentos existentes

Exemplo de Cálculo:

- Loja: 09:00 - 18:00, intervalo 30 min
- Serviço: 60 minutos
- Agendamentos existentes: 10:00-11:00, 14:00-15:00
- Horários disponíveis: 09:00, 11:00, 11:30, 12:00, 12:30, 13:00, 13:30, 15:00, 15:30, 16:00, 16:30, 17:00

5.6 Autorização

- Clientes: podem ver apenas seus próprios agendamentos
- Prestadores: podem ver agendamentos de sua loja
- Apenas o dono pode editar/cancelar/deletar

Estrutura de Dados

Entidades Principais

User (Usuário)

```
{
  id: string;              // UUID
  name: string;             // Mínimo 2 caracteres
  email: string;            // Único, formato email
```

```
  password: string;          // Hash bcrypt (nunca retornado)
  type: UserType;            // 'cliente' | 'prestador'
  cpf?: string;             // Obrigatório para PRESTADOR
  phone?: string;           // Obrigatório para CLIENTE
  createdAt: Date;
  updatedAt: Date;
}
```

Store (Loja)

```
{
  id: string;                       // UUID
  name: string;                      // Mínimo 2 caracteres
  userId: string;                    // UUID do PRESTADOR
  workingHours: WorkingHours[];      // Array de 7 dias
  location: Location;
  appointmentInterval: number;       // 5, 10, 15 ou 30 minutos
  imageUrl?: string;
  createdAt: Date;
  updatedAt: Date;
}
```

Service (Serviço)

```
{
  id: string;               // UUID
  title: string;            // Mínimo 2 caracteres
  description: string;      // Mínimo 10 caracteres
  price: number;            // Mínimo 0.01
  durationMinutes: number; // 1-1440 minutos
  storeId: string;         // UUID da loja
  imageUrl?: string;
  createdAt: Date;
  updatedAt: Date;
}
```

Appointment (Agendamento)

```
{
  id: string;                    // UUID
  userId: string;                 // UUID do CLIENTE
  storeId: string;                // UUID da loja
  serviceId: string;              // UUID do serviço
  appointmentDate: Date;          // Data/hora futura
  status: AppointmentStatus;      // pending | confirmed | completed | cancelled
  notes?: string;                 // Máximo 500 caracteres
  createdAt: Date;
```

```
  updatedAt: Date;
}
```

Relacionamentos

- **User** → **Store**: Um PRESTADOR tem uma loja (1:1)
- **Store** → **Service**: Uma loja tem muitos serviços (1:N)
- **User** → **Appointment**: Um CLIENTE tem muitos agendamentos (1:N)
- **Store** → **Appointment**: Uma loja tem muitos agendamentos (1:N)
- **Service** → **Appointment**: Um serviço tem muitos agendamentos (1:N)

Fluxos Principais

1. Fluxo de Cadastro e Autenticação

```
1. Usuário acessa tela de registro
2. Seleciona tipo (CLIENTE ou PRESTADOR)
3. Preenche dados obrigatórios:
   - CLIENTE: nome, email, senha, telefone
   - PRESTADOR: nome, email, senha, CPF
4. Sistema valida dados
5. Se válido:
   - Cria usuário no banco
   - Gera token JWT
   - Retorna token e dados do usuário
   - Redireciona conforme tipo:
     * CLIENTE → /cliente/busca
     * PRESTADOR → /prestador/home
6. Se inválido: retorna erro de validação
```

2. Fluxo de Criação de Loja (PRESTADOR)

```
1. PRESTADOR acessa configurações
2. Preenche dados da loja:
   - Nome
   - Horários de funcionamento (7 dias)
   - Localização completa
   - Intervalo de agendamento
   - Imagem (opcional)
3. Sistema valida:
   - Usuário é PRESTADOR
   - Usuário não tem loja ainda
   - Horários válidos
   - Localização válida
4. Se válido: cria loja e associa ao usuário
5. Se inválido: retorna erro
```

3. Fluxo de Criação de Serviço (PRESTADOR)

```
1. PRESTADOR acessa "Meus Serviços"
2. Clica em "Adicionar Serviço"
3. Preenche dados:
   - Título
   - Descrição
   - Preço
   - Duração
   - Imagem (opcional)
4. Sistema:
   - Identifica loja do usuário (via token)
   - Valida dados
   - Cria serviço associado à loja
5. Serviço fica disponível para agendamento
```

4. Fluxo de Agendamento (CLIENTE)

```
1. CLIENTE busca lojas
2. Seleciona uma loja
3. Visualiza serviços disponíveis
4. Seleciona um serviço
5. Sistema calcula horários disponíveis:
   a. Busca horário de funcionamento da loja
   b. Busca intervalo de agendamento
   c. Busca duração do serviço
   d. Busca agendamentos existentes
   e. Calcula slots disponíveis
6. CLIENTE seleciona data e horário
7. CLIENTE adiciona observações (opcional)
8. Sistema valida:
   - Horário está disponível
   - Horário está dentro do funcionamento
   - Não há conflitos
9. Se válido: cria agendamento com status PENDING
10. Se inválido: retorna erro
```

5. Fluxo de Visualização de Agendamentos

```
CLIENTE:
1. Acessa "Meus Agendamentos"
2. Sistema busca agendamentos do usuário autenticado
3. Exibe lista ordenada por data

PRESTADOR:
1. Acessa dashboard/home
2. Sistema busca agendamentos da loja do prestador
3. Exibe lista ordenada por data
```

6. Fluxo de Cancelamento

```
1. Usuário acessa agendamento
2. Clica em "Cancelar"
3. Sistema valida:
   - Usuário é dono do agendamento
   - Status não é CANCELLED ou COMPLETED
4. Se válido:
   - Atualiza status para CANCELLED
   - Horário fica disponível novamente
5. Se inválido: retorna erro
```

Segurança e Autenticação

Autenticação JWT

- **Algoritmo**: HS256
- **Expiração**: 24 horas (configurável via JWT_EXPIRES_IN)
- **Secret**: Configurado via JWT_SECRET (deve ser alterado em produção)
- **Payload**: Contém user ID no campo `sub`

Headers de Autenticação

```
Authorization: Bearer <jwt-token>
```

Proteção de Endpoints

- **Públicos**: Login, Signup, Listar lojas, Listar serviços, Horários disponíveis
- **Protegidos**: Todos os outros endpoints requerem JWT válido
- **Guards**: JwtAuthGuard valida token em endpoints protegidos

Hash de Senhas

- **Algoritmo**: bcrypt
- **Rounds**: 10
- **Armazenamento**: Hash nunca é retornado em respostas

Validação de Dados

- **DTOs**: Todos os dados de entrada são validados com class-validator
- **Sanitização**: class-transformer remove campos não permitidos
- **Tipos**: TypeScript garante type safety em tempo de compilação

---

API e Endpoints

Base URL

- **Desenvolvimento**: `http://localhost:3000`
- **Produção**: Configurável via variáveis de ambiente

Endpoints Principais

Autenticação

- `POST /auth/signup` - Cadastro de usuário
- `POST /auth/login` - Login

- `POST /auth/request-password-reset` - Solicitar reset de senha
- `POST /auth/confirm-password-reset` - Confirmar reset com código
- `POST /auth/reset-password` - Redefinir senha

Usuários

- `GET /users` - Listar usuários
- `GET /users/:id` - Buscar usuário
- `POST /users` - Criar usuário
- `PUT /users/:id` - Atualizar usuário
- `DELETE /users/:id` - Deletar usuário

Lojas

- `GET /stores` - Listar lojas (com busca opcional)
- `GET /stores/:id` - Buscar loja
- `GET /stores/user/:userId` - Buscar loja por usuário
- `POST /stores` - Criar loja (multipart/form-data)
- `PUT /stores/:id` - Atualizar loja (multipart/form-data)
- `DELETE /stores/:id` - Deletar loja

Serviços

- `GET /services` - Listar todos os serviços
- `GET /services/my-services` - Meus serviços (autenticado)
- `GET /services/store/:storeId` - Serviços de uma loja
- `GET /services/:id` - Buscar serviço
- `POST /services` - Criar serviço (autenticado, multipart/form-data)
- `PUT /services/:id` - Atualizar serviço (autenticado, multipart/form-data)
- `DELETE /services/:id` - Deletar serviço (autenticado)

Agendamentos

- `GET /appointments` - Meus agendamentos (autenticado)
- `GET /appointments/store/:storeId` - Agendamentos da loja (autenticado)
- `GET /appointments/available-slots/:storeId/:serviceId?date=YYYY-MM-DD` - Horários disponíveis (público)
- `GET /appointments/:id` - Buscar agendamento (autenticado)
- `POST /appointments` - Criar agendamento (autenticado, CLIENTE)
- `PUT /appointments/:id` - Atualizar agendamento (autenticado)
- `POST /appointments/:id/cancel` - Cancelar agendamento (autenticado)
- `DELETE /appointments/:id` - Deletar agendamento (autenticado)

Códigos de Status HTTP

- **200 OK**: Requisição bem-sucedida
- **201 Created**: Recurso criado com sucesso
- **204 No Content**: Recurso deletado com sucesso
- **400 Bad Request**: Dados inválidos ou validação falhou
- **401 Unauthorized**: Token ausente ou inválido
- **403 Forbidden**: Usuário não tem permissão
- **404 Not Found**: Recurso não encontrado
- **409 Conflict**: Conflito (ex: email duplicado, loja já existe)

Documentação da API

A documentação completa da API está disponível em formato Swagger/OpenAPI no arquivo `swagger.yaml`.

Considerações Finais

Pontos Importantes

1. **Validação em Múltiplas Camadas**: Dados são validados no frontend, DTOs e banco de dados
2. **Segurança**: Senhas hasheadas, tokens JWT, validação de autorização
3. **Escalabilidade**: Arquitetura modular permite fácil expansão
4. **Manutenibilidade**: Código organizado seguindo padrões e boas práticas
5. **Type Safety**: TypeScript garante tipos em todo o sistema

Melhorias Futuras

- Notificações push para agendamentos
- Sistema de avaliações e comentários
- Pagamento integrado
- Calendário visual para prestadores
- Relatórios e estatísticas
- Integração com mapas para localização

Versão do Documento: 1.0
Data: 2024
Autor: Sistema de Agendamento TCC