MedClinic MVP - Especificação Consolidada

🎯 MedClinic MVP - Especificação Consolidada

Data: 24 de Janeiro de 2026
Versão: 1.0
Status: ✅ Pronto para Implementação

📋 Índice

Escopo Geral

Visão do Produto

- **Nome:** MedClinic MVP
- **Objetivo:** Validar fluxo de agendamento de consultas com pagamento mock
- **Usuários Alvo:** Pacientes, Médicos, Recepcionistas, Administradores
- **Tipo de Consulta:** Presencial apenas
- **Duração do MVP:** ~2-3 semanas de desenvolvimento

Stack Tecnológico

```
Backend:  Node.js (v20+) + Express + TypeScript + SQLite3
Frontend: HTML5 + CSS3 + TypeScript (Vanilla, sem frameworks)
Testes:   Mocha + Chai (>80% cobertura obrigatória)
Deploy:   Railway/Render (backend), GitHub Pages (frontend)
```

Foco Técnico

- **70%:** Backend robusto, regras de negócio precisas, testes completos
- **30%:** Frontend funcional e visual

Decisões por Bloco

🔐 AUTENTICAÇÃO

Decisão	Implementação
Registro	Público (apenas pacientes). Admin cria outros roles
Email Confirmation	❌ Não
Senha	8+ caracteres, 1 maiúscula, 1 minúscula, 1 número
CPF	Obrigatório, apenas formato (XXX.XXX.XXX-XX), SEM validação de dígitos
Telefone	Opcional
Login	Email + Senha
MFA/2FA	❌ Não
"Esqueci Minha Senha"	❌ Não (admin reseta)
Logout	✅ Endpoint simples
Token	JWT em HttpOnly Cookie (24h expiração)

👥 USUÁRIOS

Rota	Método	Quem Acessa	Observação
`GET /api/v1/:clinic_id/users`	GET	clinic_admin, receptionist (filtrado), system_admin	Paginação obrigatória
`GET /api/v1/:clinic_id/users/:id`	GET	Próprio + Profissional (pacientes atendidos) + Admin	Retorna professional_details se médico
`PUT /api/v1/:clinic_id/users/:id`	PUT	Próprio (nome/email/telefone) + Admin	Admin pode atualizar tudo exceto role/password
`DELETE /api/v1/:clinic_id/users/:id`	DELETE	system_admin + clinic_admin	Soft delete (desativa). Valida consultas pendentes

👨‍⚕️ PROFISSIONAIS

Rota	Método	Observação
`GET /api/v1/:clinic_id/professionals`	GET	Pública. Filtros: specialty, name. Paginação.
`GET /api/v1/:clinic_id/professionals/:id/availability`	GET	Slots de 50min. Próximos 7 dias (default) ou custom.
`POST /api/v1/:clinic_id/professionals/:id/availability`	POST	Médico cadastra seus horários. Sem sobreposição.
`GET /api/v1/:clinic_id/professionals/:id/commissions`	GET	Médico vê suas. Admin vê qualquer um. Status: pending/paid.

Configuração de Durações:

- ✅ Hardcoded: **Todos têm 50 minutos** (simplificação MVP)

Horários Seed:

- Segunda a Sexta: 09:00-12:00 e 14:00-18:00 (para todos)

🗓️ AGENDAMENTOS (Core)

Rota	Método	Permissões	Validações MVP
`GET /api/v1/:clinic_id/appointments`	GET	Paciente (seus) / Médico (seus) / Recepcionista (todos) / Admin (todos)	Filtros: statu professiona patient_id, date, upcoming. Paginação.
`GET /api/v1/:clinic_id/appointments/:id`	GET	Paciente (seu) / Médico (seu) / Recepcionista (qualquer) / Admin (qualquer)	Retorna da completos

Rota	Método	Permissões	Validações MVP
`POST /api/v1/:clinic_id/appointments`	POST	Paciente + Recepcionista (em nome de paciente)	**RN-01:** Hor disponível **R 02:** Antecedênc 2h presenci **RN-03:** Máx dias **RN-04** Sem duplicação mesmo dia **Response:** Inclui invoic com split de pagamento (mock)
`DELETE /api/v1/:clinic_id/appointments/:id`	DELETE	Paciente (seu) + Recepcionista + Admin	>24h: 100% reembolso. <24h: 70% reembolso. Avisa na response.
`POST /api/v1/:clinic_id/appointments/:id/reschedule`	POST	Paciente (seu) + Recepcionista + Admin	Valida novo slot. SEM ta (<24h taxa Fase 2).

Status de Agendamento no MVP:

- `scheduled` (criado)
- `confirmed` (após pagamento bem-sucedido)
- `cancelled_by_patient` (cancelado)
- `cancelled_by_clinic` (cancelado por admin)

Fora do MVP:

- `/checkin`, `/start`, `/complete`, `/no-show`

🧪 EXAMES (Simplificado)

Rota	Método	Obs
`GET /api/v1/:clinic_id/exams`	GET	Paciente (seus) / Médico (que solicitou) / Lab Tech + Admin (todos)

Rota	Método	Obs
`GET /api/v1/:clinic_id/exams/:id`	GET	Detalhes do exame
`POST /api/v1/:clinic_id/exams`	POST	Médico solicita. Campo: `exam_name` (texto livre, SEM dropdown)

Fora do MVP:

- Upload de resultado (lab_tech)
- Release de resultado (liberar para paciente)
- Agendamento de coleta
- PDF de resultado

💊 PRESCRIÇÕES (Simplificado)

Rota	Método	Obs
`GET /api/v1/:clinic_id/prescriptions`	GET	Paciente (suas) / Médico (criadas por ele)
`POST /api/v1/:clinic_id/prescriptions`	POST	Médico cria. MVP: apenas `medication_name` (texto livre)

Fora do MVP:

- Dosagem estruturada, frequência, duração
- PDF com assinatura digital
- Prescrição controlada

💳 PAGAMENTOS (Mock)

Implementação	Detalhes
Gatilho	Automático ao criar agendamento
Taxa Sucesso	80% aprovado, 20% falha (randomizado)
Dados Cartão	❌ Não pede no MVP
Parcelamento	❌ Não. Apenas 1x
MDR	3.79% (desconta do bruto)
Split	60% profissional, 35% clínica, 5% sistema

Implementação	Detalhes
Response	Apenas mensagem sucesso/falha + invoice
Fora do MVP	Comprovante PDF, email, endpoint de refund

📊 COMISSÕES

Feature	Status
Visualização de comissão (GET endpoint)	✅ MVP
Cálculo automático (60% do líquido)	✅ MVP
Relatório mensal	❌ Fase 2
Cron job de pagamento	❌ Fase 2
Dashboard de admin	❌ Fase 2

✅ VALIDAÇÕES

Client-side (Frontend):

- Email (regex padrão)
- CPF (apenas formato XXX.XXX.XXX-XX)
- Telefone (formato (XX) XXXXX-XXXX)
- Senha (8+, 1 maiúscula, 1 minúscula, 1 número)
- Datas (futuro)

Server-side (Backend - Obrigatório):

- ✅ Tudo do client-side
- ✅ Lógica de negócio (horários, conflitos, antecedência)
- ✅ Autorização por role
- ✅ Integridade referencial

Error Reporting:

- Campo com borda vermelha + mensagem legível

🔒 SEGURANÇA & AUTORIZAÇÃO

Controle	Implementação
Paciente acessa dados de outro	❌ Backend rejeita (403)
Profissional vê comissão de outro	❌ Backend rejeita (403)
Recepcionista deleta usuário	❌ Backend rejeita (403). Apenas admin
Soft delete	✅ Usuarios desativados, não removidos
LGPD	Apenas desativar em vez de deletar

🌱 DADOS INICIAIS (Seeds)

```
1 × system_admin       (admin@medclinic.dev / Admin@123)
1 × clinic_admin       (gestor@clinica.com / Gestor@123)
1 × receptionist       (recepcao@clinica.com / Recepcao@123)
3 × health_professional:
    - Cardiologia     (cardio@clinica.com / Profissional@123)
    - Psicologia      (psico@clinica.com / Profissional@123)
    - Nutrição        (nutri@clinica.com / Profissional@123)
5 × patient            (maria@, joao@, ana@, pedro@, julia@ /
Paciente@123)

Profissionais: Seg-Sex, 09:00-12:00 e 14:00-18:00
Catálogo Exames: Vazio (admin cria depois)
```

Estrutura Técnica

Pastas Backend

```
src/
├── app.ts
├── server.ts
├── config/
│   └── database.ts
├── database/
│   ├── schema.sql
│   └── seed.ts
├── controllers/
│   ├── AuthController.ts
│   ├── UserController.ts
```

```
│   ├── ProfessionalController.ts
│   └── AppointmentController.ts
├── services/
│   ├── AuthService.ts
│   ├── UserService.ts
│   ├── ProfessionalService.ts
│   ├── AppointmentService.ts
│   └── PaymentMockService.ts
├── repositories/
│   ├── UserRepository.ts
│   ├── ProfessionalRepository.ts
│   ├── AvailabilityRepository.ts
│   ├── AppointmentRepository.ts
│   ├── TransactionRepository.ts
│   └── CommissionSplitRepository.ts
├── models/
│   ├── User.ts
│   ├── Appointment.ts
│   ├── Professional.ts
│   └── ...
├── middlewares/
│   ├── authMiddleware.ts
│   └── errorHandler.ts
├── utils/
│   ├── validators.ts
│   └── paymentMock.ts
└── __test__/
    ├── auth.test.ts
    ├── appointment.test.ts
    └── ...
```

Pastas Frontend

```
frontend_src/
├── main.ts
├── styles/
│   ├── global.css
│   ├── forms.css
│   └── components.css
├── types/
│   └── api.ts
├── services/
```

```
│   └── api.ts
├── components/
│   ├── Modal.ts
│   ├── Toast.ts
│   └── Form.ts
└── pages/
    ├── Login.ts
    ├── RegisterPatient.ts
    ├── DashboardPatient.ts
    ├── DashboardDoctor.ts
    └── DashboardAdmin.ts
```

Padrão de Response API

Sucesso:

```
{
  "success": true,
  "data": { /* payload */ },
  "pagination": { "total": 10, "page": 1, "pageSize": 20 },
  "message": "Descrição"
}
```

Erro:

```
{
  "success": false,
  "error": {
    "code": "ERROR_CODE",
    "message": "Mensagem legível",
    "statusCode": 400,
    "field": "campo" // opcional
  }
}
```

Status HTTP

Status	Quando
200	Sucesso geral
201	Recurso criado
400	Validação falhou
401	Não autenticado
403	Não autorizado
404	Não encontrado
409	Conflito (email duplicado, horário ocupado)
500	Erro servidor

Autenticação Global

- **Método:** JWT em HttpOnly Cookie
- **Exclusões:** `/api/v1/:clinic_id/auth/register`, `/api/v1/:clinic_id/auth/login`, `GET /api/v1/:clinic_id/professionals`
- **Payload:** `{ id, email, role, iat, exp }`
- **Expiração:** 24 horas
- **CORS:** `http://localhost:3001`

Próximos Passos

✅ Fase 1: Fundação

1. Criar `schema.sql` (DDL com todas as tabelas)
2. Implementar `config/database.ts` (Singleton + conexão)
3. Script de `seed.ts` (dados iniciais)
4. Setup Express + TypeScript + CORS

✅ Fase 2: Autenticação

1. `AuthService` (hash, JWT, validação)
2. `AuthController` (rotas de login/register/logout/profile)
3. `authMiddleware` (verificar token + role)
4. Testes unitários

✅ Fase 3: Usuários

1. `UserRepository` + `UserService` + `UserController`
2. Validações (CPF formato, email, senha)
3. Autorização por role
4. Testes

✅ Fase 4: Profissionais

1. `ProfessionalDetailsRepository` + `AvailabilityRepository`
2. Listar profissionais + disponibilidade
3. Cadastro de horários
4. Comissões (visualização)

✅ Fase 5: Agendamentos

1. `AppointmentRepository` + `AppointmentService`
2. Todas as validações (RN-01, 02, 03, 04)
3. Integração com PaymentMockService
4. Cancelamento + Reagendamento
5. Testes extensivos

✅ Fase 6: Frontend Vanilla

1. Estrutura básica (HTML + CSS global)
2. API Helper (fetch com cookie)
3. Páginas: Login, Register, Dashboard (paciente/médico/admin)
4. Formulários com validação client-side
5. Feedback visual (toast, modals)

✅ Fase 7: Testes & Deploy

1. Testes unitários (>80% cobertura)
2. README com instruções setup
3. Deploy backend (Railway/Render)
4. Deploy frontend (GitHub Pages)

📚 Documentação Gerada

Documento	Status	Descrição
`API_ROUTES_DOCUMENTATION.md`	✅	Especificação técnica de TODAS as rotas MVP
`MVP_SPECIFICATION.md`	✅	Este arquivo - decisões consolidadas
`database_schema.sql`	⏳ Próximo	DDL com tabelas e índices
`CODE_STRUCTURE.md`	⏳ Próximo	Organização de código + padrões

🚀 Status

MVP Specifications: ✅ 100% DEFINIDO E APROVADO

Pronto para começar a implementação!